针对古代玻璃制品的成分分析与鉴别的研究

摘要

本文采用了**方差分析**、**多元线性回归**等多种统计学方法对相关数据进行了研究，得到了**玻璃种类分类规律**、**化学含量统计规律**等结果。

对于问题一，本文首先将颜色划分为有序变量，其余指标划分成为分类变量。对表面风化与玻璃类型和纹饰的关系，本文对表单 1 数据采用**列联表分析**的方式进行**卡方检验**；对表面风化与颜色的关系，采用 **Rank-Biserial 秩相关分析**进行相关性检验。结论为**玻璃类型和颜色与表面风化存在显著相关关系，与纹饰则没有显著相关关系**。

接着，本文结合附件中表单 1 和 2 的数据，将数据分为高钾风化、铅钡风化、高钾未风化、铅钡未风化四类，对四类数据分别做**正态性检验**，并对整体做**方差齐性检验**，针对数据本身的不同分布特点，对 14 种化学成分分别进行**多因素方差分析**或**独立样本 t 检验**和**非参数检验**，结果见 **5.2.5~5.2.12**。

最后，对于两种玻璃类型的风化点，分别求出 14 种化学成分含量两两之间的皮尔逊相关系数。对于某一化学成分，筛选出与其相关系数最大的另外三个化学成分，与风化程度共同作为自变量，该化学成分含量作为因变量，进行**多元线性回归**。再通过改变风化程度变量的取值为无风化，代入回归结果解得各化学成分在风化前的预测值，结果见表 14。

对于问题二，为了消除风化对化学成分含量的影响，本文利用问题一中的预测结果将玻璃各化学成分统一转化为风化前的水平，采用**逻辑回归模型**，求解出逻辑回归的表达式见式 (12)。接着，本文先以所有化学成分为标准采用**系统聚类**进行粗聚类，得到**粗聚类结果**，再采用**方差分析**检验各化学成分在各组中的差异性，去除掉组间无显著差异的化学成分。然后将剩余成分作为标准重新进行系统聚类，得到**最终聚类结果**，即将高钾玻璃分为 **2 个亚类**，铅钡玻璃分为 **3 个亚类**，结果见表 17。本文从数学效果与现实意义两个角度分析和说明了聚类结果的合理性。本文还定义了各指标的**抗干扰能力**，用于分析模型对不同指标输入的敏感性。每次调整单个化学成分的输入值，再重新聚类，检测聚类结果是否产生显著变化，从而得到抗干扰能力的大小。结论为**高钾玻璃聚类结果对二氧化硅含量较敏感**，**铅钡玻璃对二氧化硅和氧化铅含量较敏感**。

对于问题三，本文首先将表单 3 的数据代入问题二中的逻辑回归方程，得到 8 个样品的分类，结果见表 21。接着，本文采用 **Logloss 损失函数**来作为评价分类效果的指标，从逻辑回归模型本身的参数设置和输入的数据两个方面进行敏感性分析。结论为**分类结果对两个参数的选择有一定的敏感性**；对于输入数据，**分类结果对 14 种化学成分的输入都不显著敏感**，说明了模型的稳定性。

对于问题四，本文首先对 14 种化学成分进行两两之间的**皮尔逊相关系数检验**，分别得到高钾和铅钡玻璃内部呈现显著相关关系的组合，结果如图 15a 和图 15b所示。接着，本文对两种类型的玻璃的相关系数进行 **Fisher z 变换**和**相关系数 z 检验**，比较了高钾玻璃和铅钡玻璃之间的化学成分关联关系的差异性，得到了 27 组在不同类别之间关联关系有显著差异的化学成分组合，结果见图 16。最后，本文对这 27 组具有差异性的组合做了进一步分类，分析了造成差异的三种可能，并结合具体的组合分别进行了解释。

本文利用多种统计学模型对问题进行了细致的求解，并特别关注了对结果敏感性的分析，得出的结论大都在现实生活中有实际含义，具有很好的可解释性。

关键字：古代玻璃 方差分析 多元线性回归 逻辑回归 敏感性分析 相关系数 z 检验

一、 问题重述

1.1 问题背景

我国古代玻璃制造在吸取西亚和埃及地区的制造技术之后，通过就地取材制作出的玻璃制品，外观与西亚和埃及地区的珠形饰品相似，但化学成分不同。玻璃的主要化学成分是二氧化硅，由于其熔点较高，在炼制时需要添加助熔剂。根据使用的助熔剂不同，可以获得不同类型的玻璃，我国古代玻璃主要以铅钡玻璃和高钾玻璃为主。而由于古代玻璃极易受埋葬环境的影响而风化，其内部化学成分和比例会发生变化，从而影响对其类别的正确判断。

1.2 数据集

已知现有一批古代玻璃制品的相关数据，且已经将这些文物样品分成高钾玻璃和铅钡玻璃两种类型。附件表一是该批文物的分类信息，表二是相应的主要成分所占总体的比例，其各成分比例求和应该为 100%，但因检测手段等原因可能导致其各成分比例的累加求和非 100% 的情况。根据题目要求将成分比例累加求和介于 85%~105% 之间的数据视为有效数据，其余数据为无效数据。

1.3 问题提出

根据上述的背景和数据，建立数学模型解决以下问题：

问题一：分析给出的玻璃文物的表面风化与其玻璃类型、纹饰和颜色的关系；结合玻璃的类型，对文物样品表面有无风化化学成分含量的统计规律进行分析，并根据风化点检测数据，预测该玻璃文物风化前的各化学成分含量。

问题二：对高钾玻璃、铅钡玻璃的分类规律进行分析；对于这两个类别分别选择合适的化学成分对其进行亚类划分，给出具体的划分方法和结果，并对分类结果的合理性和敏感性进行分析。

问题三：通过对附件表 3 中未知类别的玻璃文物的化学成分进行分析，给出其所属类别，并对分类结果的敏感性进行分析。

问题四：分析不同类型的样品的化学成分之间的关联关系，并比较不同类型的文物样品之间的化学成分关联关系的差异性。

二、 问题分析

2.1 问题一分析

通过阅读题干，可以明显看出该问可分为三个子问题，同时根据问题背景可知：数据集中可能存在无效数据，于是需要进行数据预处理。

观察数据集可以发现，题目给出的玻璃样品中存在未风化和风化两大种，而单个风化文物中又存在未风化点和严重风化点，由此本文给出树状图 1，并认为未风化文物在该题当中不存在风化点。在该假设下，本文对附件表一给出的数据进行预处理，对颜色数据存在缺失的文物样品，在分析颜色与有无风化的相关关系时将其剔除。

对于子问题一，要求分析文物的表面风化与玻璃类型、纹饰和颜色三者的关系。由于表面风化与三者之间不存在明显因果关系，因此本文选择对他们的相关关系进行分析。为此，需要首先确定这四个变量的类型，然后再根据变量类型来确定相应的分析方法，最后得出三者的相关关系并进行分析。

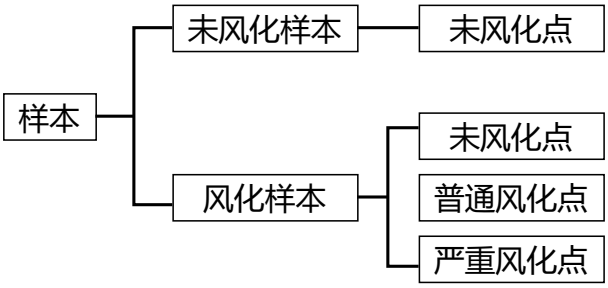


图 1　风化类别划分树状图

对于子问题二，要求结合文物的玻璃类型，分析其表面有无风化对文物内化学成分含量的影响的统计规律。由于要对文物化学成分含量进行分析，根据题目对给出的数据集的描述，需要先对数据进行预处理，筛选出有效数据；再将附件表一和表二进行信息整合，得到新表。对于此类问题，多因素方差分析是最好的选择，其次是独立样本 t 检验或非参数检验。于是本文在新表的基础上，对每一种化学成分而言，将数据分成四大部分（有风化高钾、有风化铅钡、无风化高钾、无风化铅钡），观察统计图像，若无显著规律，则单独分析，若有显著规律，则对四个部分的数据分别进行方差齐性分析，如果满足则进行正态分布检验，若再满足则进行多因素方差分析；对于不满足方差齐性检验的情况，根据图像规律在表面风化和玻璃类型两个因素之间选一个进行组间正态分布检验，若满足，则进行独立样本 t 检验，若不满足，则进行非参数检验，由此得到一级结论。再通过观察图像，基于第一次选择的结果（例如玻璃类型）选择单个因素（例如高钾类型）进行内部分析（例如分析有无风化），同样进行正态分布检验，满足做独立样本 t 检验，不满足做非参数检验，由此得到二级结论。

对于子问题三，根据题意可以判断该题为回归预测题，需要得到显示的方程用于预测表面风化文物未风化前的内部化学成分的含量。对于该题，本文将风化程度和不同化学成分之间的相互影响同时纳入考虑，对预测同一种化学成分在不同的玻璃类型样本内的含量采用不同的预测方程，由此得到附件二中风化样本点未风化前的化学成分含量。

2.2 问题二分析

通过阅读题干，可以明确题目可划分成三个子问题。

对于子问题一，要求根据数据分析高钾玻璃和铅钡玻璃的分类规律。这实际上是一个二分类问题，由于目前对二分类问题已有较为成熟的方法，一般采用逻辑回归、SVM、KNN 等传统算法或 XGBoost、随机森林等集成算法。对于该问题，题目给出的数据量较少，不适用于需要大量数据进行训练的集成算法，故本文选用传统算法对该问题进行求解。在传统算法中，逻辑回归可以较为简单的获得一个显性的表达式，模型较为简单，且可以得到某一样本判定为某一类的概率，便于后续的分析，故本文采用逻辑回归进行该子问题的求解。

对于子问题二，要求对不同的玻璃类别选择化学成分进行亚类划分，给出具体划分方法和结果。由于题目没有明确是否可以使用表面风化这一指标进行亚类划分，本文根据控制变量的思想，将附件表二中所有的风化样本点的数据利用问题一的预测模型转化成风化前的数据，并将得到的风化前的数据与附件表二中的未风化点的数据整合成新表，用于该题的讨论和分析。由于该问题要求进行亚类划分，对于此类问题一般采用聚类方法解决，但是本文有较多的聚类指标，需要进行筛选以剔除无用指标，最后得到合适的指标进行聚类以确定最终的亚类划分。

对于子问题三，要求对子问题二当中的亚类划分的结果进行合理性和敏感性进行分析。本文将其分为合理性和敏感性分别进行分析。

对于合理性，本文将其分为数值层面和实际生活层面进行分析。对于数值层面，由

于本文采用的是聚类方法，已有相应的合理性分析指标（轮廓系数）；对于实际生活层面，本文通过查阅文献获得对应化学成含量变化对于玻璃的理化特性产生的影响。将不同亚类之间化学成分含量变化对应到玻璃的理化特性上，借此分析聚类结果的合理性。

对于敏感性，本文通过选定每个亚类中对于某一化学成分而言最接近其类内平均值的 1 或 3 个样本点作为该亚类的 “锚点”，用于表示该亚类对于这一化学成分而言的定位特征，通过逐一改变该亚类中除 “锚点” 以外的样本点的该化学成分的含量，在满足控制变量的原则下重新进行聚类，在新的聚类结果中定位该样本点与原锚点是否划分在同一类中。对不在同一类的样本点进行计数，最后的到对于该化学成分而言，分类结果的敏感性。

2.3 问题三分析

通过阅读题干，可以将原题划分成两个子问题。

对于子问题一，要求利用附表三中的未知类别玻璃文物的化学成分，对这些文物的玻璃类别进行分类。由于在问题二中已经对玻璃类别的分类规律进行建模分析，在该问当中可以直接利用前面得到的分类模型代入相关数据进行类别的求解。

对于子问题二，要求分析子问题一分类结果的敏感性。由于对于一个分类问题，只需要有分类模型和和输入的数据，即可根据数据给出分类结果，所以本文对分类模型本身的敏感性和分类模型对输入数据的敏感性分别进行分析，以求对分类结果进行更加全面的分析。

2.4 问题四分析

阅读题干，将问题分为两个部分。

对于子问题一：要求分析不同玻璃类型的文物内部化学成分之间的关联关系。由于本文在问题二的风化前化学成分预测时使用了 Pearson 相关系数检验来获得化学成分之间的相关程度，用以筛选影响最大的化学成分用于预测；在该问当中本文继续使用前文计算得到的 Pearson 相关系数矩阵，并进一步进行 Pearson 相关系数检验，以此刻画化学成分的关联关系。

对于子问题二：要求比较不同类别，相同的一组化学成分之间的关联关系存在的差异性。本文在子问题一的基础上，利用 Pearson 相关系数矩阵中的数据进行相关系数 z 检验，以确定不同玻璃类别的哪些化学成分组合的相关关系存在差异，再对这也存在差异的相关关系结合 Pearson 相关系数矩阵进行进一步的分析。

三、 模型假设

1、假设对未风化的玻璃文物采样点都是未风化点。

2、假设古代玻璃的鉴别仅与附件中给出的指标有关，不考虑其他因素的影响。

3、假设问题给出的文物的理化性质仅受附件中列出的化学成分的影响，不考虑文物内部其他化学成分的影响。

四、 符号说明

符号	含义	符号	含义
χ	卡方统计量	β	逻辑回归的拟合参数
r	Rank-Biserial 相关系数	J	系统聚类的剧变程度
A, B	多因素方差分析的因素	K	系统聚类的分类个数

pvalue	假设检验的 p 值	S	轮廓系数
t_1, t_2	风化程度的虚拟变量	η	指标抗干扰能力
p	预测方程的系数	$Logloss$	Logloss 值
$F(x, \beta)$	逻辑分布的累计分布函数	z	Fisher z 变换结果

五、 模型的建立和求解

5.1 问题一：相关关系分析

5.1.1 思路分析

根据问题分析，本文给出以下思路分析图 2a。本文接下来的模型建立流程就基于该流程图实现。

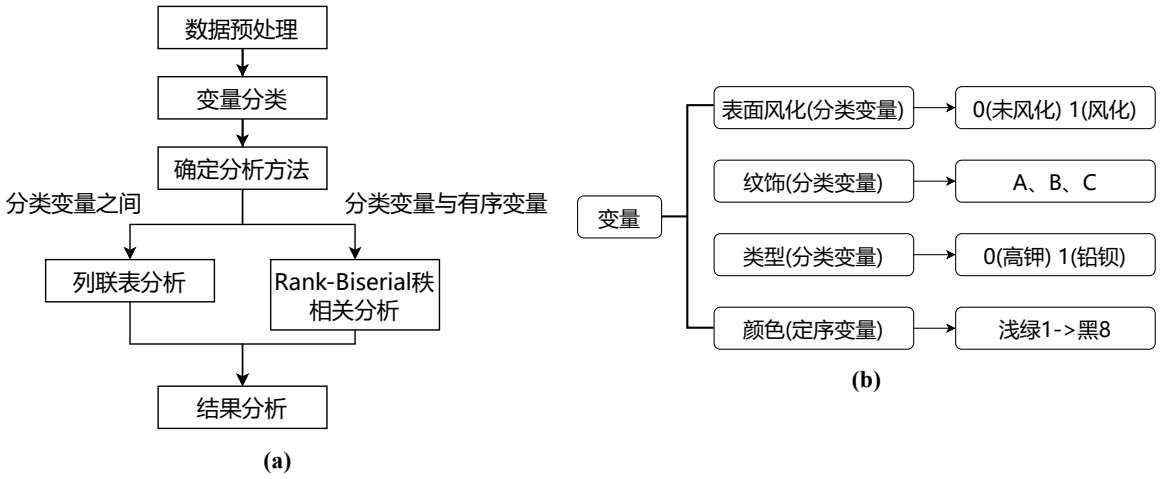


图 2　(a) 问题一第一小问思路分析图 (b) 变量类型划分

5.1.2 数据预处理

根据问题分析，本文在对玻璃类型以及纹饰的相关关系进行分析时，使用附件表二中的所有数据，在对颜色的相关关系进行分析时将颜色数据存在缺失的文物样本剔除，用剩下的样本作为分析的基础。

5.1.3 变量分类

对于表面风化、玻璃类型、纹饰和颜色这四个变量，首先根据分析可知：表面风化为自变量，其余三者为因变量。

对于四个变量的类型本文认为：表面风化、玻璃类型和纹饰这三个变量的不同值之间不存在比较或递进关系，更多的是用于体现类别关系，于是将这三个变量划分为分类变量；对于颜色，不同值之间根据光谱可以认为存在递进关系，本文认为其为有序变量。给出上图 2b，并对颜色这一有序变量的具体值根据图 3进行排序划分。

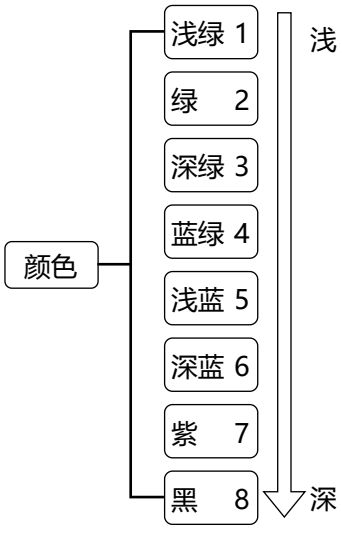


图 3　颜色分类

5.1.4 确定分析方法

由于表面风化、玻璃类型和纹饰是分类变量，对于两个分类变量之间的相关性分析通常可以采用列联表分析和相关性检验两种方法，相关性检验要求两个分类变量之间存在配对关系，而列联表分析要求每个分类变量的各个类别是互斥的、每个变量中的各个类别是互相独立的以及每一个观测值只属于联表得到的交叉表中的一个单元格。

由于表面风化与玻璃类型和纹饰都不存在配对关系，且满足列联表分析的要求，于是本文采用列联表分析的方法对表面风化与玻璃类型和纹饰分别进行相关关系的分析。

对于颜色这一有序变量，本文将图 3中的颜色深浅分类等级赋值给相应的颜色，由于表面风化为分类变量，且两者之间不存在明显因果关系，于是本文采用 Rank-Biserial 秩相关分析对两者的相关关系进行分析。

5.1.5 列联表分析

a. 建表

对于表面风化和玻璃类型之间的关系，本文建立以下表 2，对于表面风化和纹饰之间的关系，本文建立以下表 3。

表 2　表面风化与玻璃类型

表面风化 / 玻璃类型	0(无)	1(有)	合计
0(高钾)	12	6	18
1(铅钡)	12	28	40
合计	24	34	58

表 3　表面风化与纹饰

表面风化 / 纹饰	0(无)	1(有)	合计
A	11	11	22
B	0	6	6
C	13	17	30
合计	24	34	58

b. 给定原假设与备择假设

针对该问题，本文分别将 “H_0：因变量与表面风化没有关系” 作为原假设，相应的 “H_1：因变量与表面风化有关系” 作为备择假设，使用卡方独立性检验进行假设检验。其中因变量为玻璃类型和纹饰，分别进行两次假设检验。

c. 卡方独立性检验

对于表 2和表 3，将表中的行号记为 i，列号记为 j，用以下公式 (1) 计算对应表的卡方统计量。

$$\chi^2 = \sum_{i=1}^{2}\sum_{j=1}^{2} \frac{\left(n_{ij} - \frac{n_{i.}*n_{.j}}{n}\right)^2}{\frac{n_{i.}*n_{.j}}{n}} \tag{1}$$

其中，n_{ij} 表示表中单个数据，$n_{i.}$ 表示第 i 行合计，$n_{.j}$ 表示第 j 列合计，n 为行列总合计。

将表 2和表 3的数据代入上式 (1)，得到以下两个卡方统计量的计算值。

表 4　卡方统计量

玻璃类型与表面风化	纹饰与表面风化
4.9565	5.4518

此处给定显著性水平 0.05，由于玻璃类型与表面风化的卡方值 $> \chi^2_{0.05}(1) = 3.84$，此处自由度为 $1 = (\text{玻璃类型数} 2-1) \times (\text{表面风化类别} 2-1)$；纹饰与表面风化的卡方值 $< \chi^2_{0.05}(2) = 5.99$，此处自由度为 $2 = (\text{纹饰类别} 3-1) \times (\text{表面风化类别} 2-1)$。

由此给出卡方独立性检验的 p 值，见表 5。

表 5　卡方独立性检验 p 值

玻璃类型与表面风化	纹饰与表面风化
0.0195	0.0839

由此本文得出以下结论：玻璃类型与表面风化的卡方独立性检验在置信水平 95% 下可以拒绝原假设，两者有显著相关关系；纹饰与表面风化的卡方检验在置信水平 95% 下不能拒绝原假设，两者没有显著相关关系。

5.1.6 Rank-Biserial 秩相关分析

Rank-Biserial 秩相关分析常用于分析二分类变量和有序分类变量之间的相关关系，针对该问题，颜色为连续的有序变量，表面风化为无序的分类变量，适合使用 Rank-Biserial 秩相关分析。

对于 Rank-Biserial 相关系数，其计算方法为式 (2)。

$$r = 2\frac{M_1 - M_0}{n} \tag{2}$$

其中 M_1 为表面风化为 1 对应的颜色的均值，M_0 为表面风化为 0 对应的颜色的均值，n 为样本数据点的个数。

将附件表一中有效数据的颜色和表面风化的具体值代入计算，得到 p 值为 0.0127。由此可知：颜色与表面风化显著相关关系。同时本文通过对表面风化和颜色进行数据统计和可视化，得到以下气泡图 4。

图 4中的每个圆的圆心为相对应的组合，圆的大小衡量了该组合出现的次数。由该图可直观感受到，风化前存在部分颜色较浅的文物，而风化后文物的颜色普遍加深，且颜色标记为 4 和 5 的文物数量明显增多。从原题中给出的图片也可以看出风化会对文物的颜色产生影响，由此更加佐证本文得到的结论：颜色与表面风化存在显著相关关系。

5.1.7 结果分析

本文得到的结果为：玻璃类型和颜色表面风化存在显著相关关系，纹饰与表面风化不存在显著相关关系。对此结果，本文做以下分析：首先，纹饰是考古学家常常用来判别文物年代的文物特征之一，该特征应该与该文物被制造的历史时期有关系，并且不同时期的文物纹饰可能出现 “复古” 的样式 [1]，因此与表面风化的关系不明显。其次，文物的颜色也是文物鉴定专家常常用于推断文物年份的文物特征之一，往往可以根据颜色的深浅鉴定其年代的早晚 [1]。最后，玻璃类型的判定需要依靠文物内部的化学成分，而这些化学成分受风化的影响较大，所以呈现出玻璃类型与表面风化的显著相关关系。

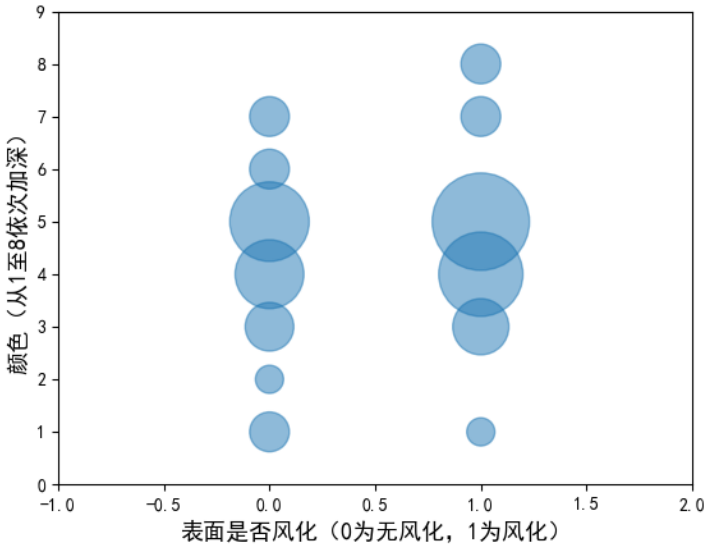


图 4　表面风化与颜色的气泡图

5.2 问题一：统计规律分析

5.2.1 思路分析

根据问题分析，本文给出以下思路分析图 5。

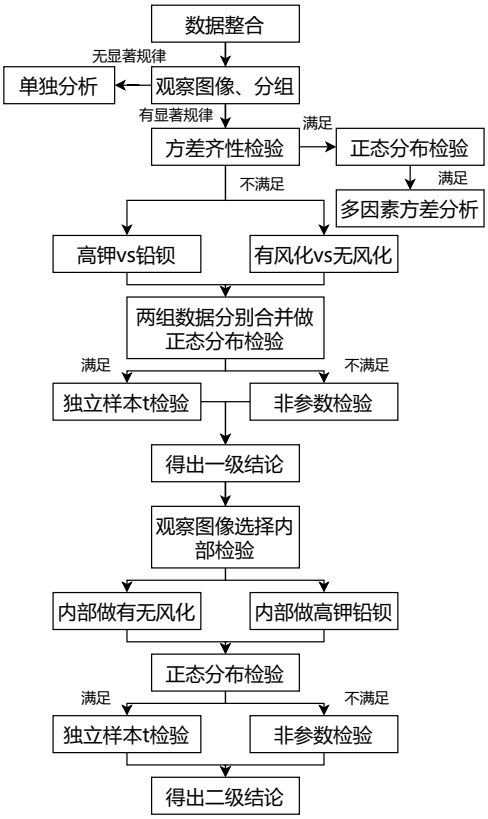


图 5　问题一第二小问思路分析图

5.2.2 数据预处理

根据问题分析可知，对该问的求解需要利用附件表二，而表二中各成分比例累加和小于 85% 和大于 105% 的数据都为无效数据，由此先对表二进行数据预处理，剔除无效数据，根据题意，对有效数据中的空白值全部取 0。

经计算附件表二中的 15 号和 17 号文物样品的内部化学成分比例累加分别为 79.47% 和 71.89%，小于 85%，据此本文将这两个文物样品的数据剔除。

由于该问只需要利用表一中的玻璃类型和表面风化两类数据，而表一中的数据只存在部分颜色数据的缺失，于是本文不需要对表一进行数据预处理。

将附件表一和表二进行联表操作，将表一中的文物编号、类型和表面风化的数据挑出，将表二中的所有有效数据挑出，对同一编号的样品，将表二的数据填在表一数据的后面，整合获得新表。对于表二中对同一文物的不同采样点数据，本文认为：同一文物的不同部分可能会因为埋藏环境不同导致其表面风化程度和内部化学成分存在较大差异，例如大型玻璃制品可能部分埋藏在土中，部分裸露在空气中，于是对于同一文物的不同采样点，本文将他们分别视为不同的文物进行处理和分析。对于表二采样点的风化程度与表一文物的风化程度不同的数据，本文统一采用表二的数据。以下表 6给出新表的格式。

表 6　新表格式

文物编号	类型	表面风化	二氧化硅 (SiO_2)	···	二氧化硫 (SO_2)
01	高钾	无风化	69.33	···	0.39
02	铅钡	风化	36.28	···	0
03_1	高钾	无风化	87.05	···	0
03_2	高钾	无风化	61.71	···	0
28	铅钡	无风化	68.08	···	0

5.2.3 观察统计图像

根据问题分析，将新表得到的数据划分成无风化高钾、无风化铅钡、有风化高钾、有风化铅钡四个部分，对表中每一个化学成分，将每一部分的文物的该化学成分含量进行排序，得到对每个化学成分的四个部分的数据统计图 6。

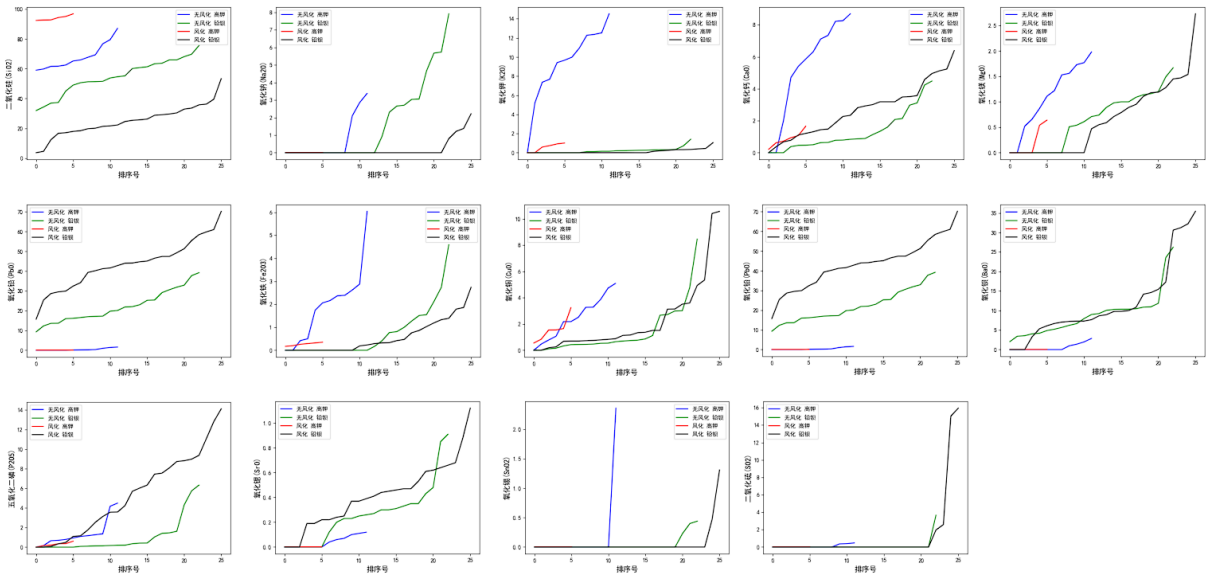


图 6　问题一第二小问数据统计图

根据图 6可知，氧化锡和二氧化硫的非零数据过少，不便于进行多因素方差分析或独立样本 t 检验等分析方法，故单独进行分析。剩下的化学成分都可以进行接下来的相关检验来确定是否能够进行多因素方差分析。

5.2.4 相关检验

根据问题分析，对新表进行数据划分，划分成四个部分之后，先进行方差齐性检验，然后进行正态分布检验，确定可以优先进行多因素方差分析的化学成分 [2]。

检验结果如下表 7，其中每个化学成分对应每个部分的正态 JB 检验 p 值以及总体的方差齐性检验 p 值如表中所示。

由表 7知，只有二氧化硅和氧化镁通过了正态分布检验和方差齐性检验，对这两个化学成分做多因素方差分析。以下对这两个化学成分与玻璃类型和表面是否分化进行多因素方差分析。

表 7　相关检验结果

化学成分	无风化高钾	无风化铅钡	有风化高钾	有风化铅钡	方差齐性检验 p 值
二氧化硅	0.3606	0.6377	0.7282	0.5942	0.0612
氧化钠	0.1823	0.0593	NAN	$3.06E-13^{***}$	0.0078^{***}
氧化钾	0.2916	0^{***}	0.6954	$6.88E-15^{***}$	$3.57E-08^{***}$
氧化钙	0.4793	0.0413^{**}	0.9036	0.6208	0.0038^{***}
氧化镁	0.6407	0.4746	0.5938	0.0922	0.1212
氧化铝	0.7226	1.2907	0.7859	0^{***}	0.5395
氧化铁	0.2513	$1.88E-06^{***}$	0.7791	0.0128^{**}	0.0943
氧化铜	0.7221	$8.58E-11^{***}$	0.6725	$8.92E-07^{***}$	0.5645
氧化铅	0.2371	0.4395	NAN	0.9974	0.0001^{***}
氧化钡	0.1801	0.0001^{***}	NAN	0.0386^{**}	0.0062^{***}
五氧化二磷	0.1042	$1.15E-05^{***}$	0.8967	0.4759	$6.79E-07^{***}$
氧化锶	0.4868	0.0349^{**}	NAN	0.516	0.003^{***}

5.2.5 多因素方差分析

本文在新表的基础之上，建立以表面风化、玻璃类型为控制变量，以化学成分（二氧化硅和氧化镁）为观测变量的多元多因素方差分析模型 [3]。

a. 明确观测变量和控制变量

由于本文要建立双因素方差分析模型，需要明确单个的水平个数。设表面是否风化为因素 A，玻璃类型为因素 B。表面风化因素共有 2 个水平 A_1, A_2，玻璃类型因素共有 2 个水平 B_1, B_2。对因素 A、B 的每一个水平的一组数据 $(A_i, B_j), (i = 1, 2; j = 1, 2)$ 只进行一次试验，得到 $L\,(= 2 \times 2)$ 个试验结果 X_{ij}。

b. 确定原假设和备择假设

检验各个因素样本的均值是否相等，如果相等，那么代表单个观测变量在控制变量改变前后没有显著差异，提出原假设为：

$$H_{0A} : \mu_{1j} = \mu_{2j}\,(j = 1, 2)\, H_{0B} : \mu_{i1} = \mu_{i2}\,(i = 1, 2) \tag{3}$$

备择假设为：

$$H_{1A} : \mu_{1j}, \mu_{2j}\text{不全相等} H_{1B} : \mu_{i1}, \mu_{i2}\text{不全相等} \tag{4}$$

c. 构建模型

针对该问题，双因素控制变量分别为：表面是否风化和玻璃类型。根据相关检验中的结果可知：数据服从正态分布，即有 X_{ij}~$N\left(\mu_{ij}, \sigma^2\right)$ $\left(\mu_{ij}\text{和}\sigma^2\text{未知}\right)$，记 $X_{ij} - \mu_{ij} = \varepsilon_{ij}$，则有 $\varepsilon_{ij} = X_{ij} - \mu_{ij}$~$N\left(0, \sigma^2\right)$，故 $X_{ij} - \mu_{ij}$ 可视作随机误差，从而建立如下数学模型：

$$\begin{cases} X_{ij} = \mu_{ij} + \varepsilon_{ij}, (i = 1, 2; j = 1, 2) \\ \varepsilon_{ij}\text{~}N\left(0, \sigma^2\right), \left(\mu_{ij}, \sigma^2\text{未知}, \varepsilon_{ij}\text{相互独立}\right) \end{cases} \tag{5}$$

引入以下变量：

$$\begin{aligned} \mu &= \frac{1}{2 \times 2} \sum_{i=1}^{2} \sum_{j=1}^{2} \mu_{ij} \\ \mu_{i\cdot} &= \frac{1}{2} \sum_{j=1}^{2} \mu_{ij}, (i = 1, 2) \\ \mu_{\cdot j} &= \frac{1}{2} \sum_{i=1}^{2} \mu_{ij}, (j = 1, 2) \\ \alpha_i &= \mu_{i\cdot} - \mu, (i = 1, 2) \\ \beta_j &= \mu_{\cdot j} - \mu, (j = 1, 2) \end{aligned} \tag{6}$$

可知，$\sum_{i=1}^{2}\alpha_i=0$，$\sum_{j=1}^{2}\beta_j=0$。其中，μ 为总平均值，α_i 为表面风化水平 A_i 的效应，β_j 为玻璃类型水平 B_j 的效应，且有 $\mu_{ij}=\mu_{ij}+\alpha_i+\beta_j$。

于是上述模型可转换成

$$\begin{cases} X_{ij}=\mu_{ij}+\varepsilon_{ij},(i=1,2;j=1,2) \\ \varepsilon_{ij}\sim N\left(0,\sigma^2\right),\left(\mu_{ij},\sigma^2\text{ 未知},\varepsilon_{ij}\text{ 相互独立}\right) \\ \sum_{i=1}^{2}\alpha_i=0,\sum_{j=1}^{2}\beta_j=0 \end{cases} \tag{7}$$

d. 模型求解

本文使用 Python 的 statsmodels 库中的 ols 和 anova_lm 函数进行双因素方差分析 [4]（具体代码见附录），即可得到结果，见表 8。

表 8　方差分析结果

观测变量	控制变量	df	sum_sq	mean_sq	F	PR(>F)
二氧化硅	表面风化	1.0	7632.7963	7632.7963	71.2579	6.0462e-12
二氧化硅	玻璃类型	1.0	15288.7402	15288.7402	142.7321	7.7134e-18
二氧化硅	表面风化: 玻璃类型	1.0	9355.2851	9355.2851	87.3387	1.6416e-13
氧化镁	表面风化	1.0	0.8527	0.8527	2.1882	0.1441
氧化镁	玻璃类型	1.0	0.1224	0.1224	0.3141	0.5772
氧化镁	表面风化: 玻璃类型	1.0	2.3974	2.3974	6.1519	0.0158

e. 结论分析

根据上表 8的 PR 值可知：表面风化和玻璃类型对文物样品内的二氧化硅含量有显著影响，且两者在对二氧化硅的含量的影响上有交互作用；表面风化和玻璃类型对氧化镁含量没有显著影响，表面风化和玻璃类型的交互效应对氧化镁含量有一定的影响但不显著。

5.2.6 组间比较选择

根据图 6，本文通过观察单个统计图的曲线增长趋势和增长幅度，对不能进行多因素方差分析的化学成分进行如下的组间比较选择。

表 9　组间比较选择

化学成分	比较选择	正态检验 p 值 (组 1)	正态检验 p 值 (组 2)	分析方法
氧化钠	有无风化	0.0	0.0003	非参数检验
氧化钾	高钾铅钡	0.3866	0.0	非参数检验
氧化钙	高钾铅钡	0.3489	0.1027	独立样本 t 检验
氧化铝	高钾铅钡	0.5980	0.0	非参数检验
氧化铁	高钾铅钡	0.0059	0.0	非参数检验
氧化铜	高钾铅钡	0.5677	0.0	非参数检验
氧化铅	高钾铅钡	0.0025	0.3476	非参数检验
氧化钡	高钾铅钡	0.0004	1.134e-06	非参数检验
五氧化二磷	高钾铅钡	0.0002	0.0099	非参数检验
氧化锶	高钾铅钡	0.1420	0.0862	独立样本 t 检验

表 9中根据组间比较选择，将原本的四大部分数据合并成两大部分，分别进行正态分布检验，若两组都服从正态分布，使用独立样本 t 检验进行分析，否则使用非参数检验。

5.2.7 组间独立样本 t 检验

独立样本 t 检验用于分析定类数据与定量数据之间的差异情况，且要求定类变量为二分类变量 [2]，对于本题而言，即为分析有无风化（或玻璃类型为高钾还是铅钡）对文物样品内化学成分（氧化钙和氧化锶）含量的差异情况。

本文使用 Pyhton 中的 ttest_ind 函数对表 9中的独立样本 t 检验进行求解，得到以下结果。

对于氧化钙：独立样本 t 检验的 p 值为 0.0043^{***}，可认为不同玻璃类型对于文物内部氧化钙含量在 99% 的置信水平下呈现显著相关性，意味着不同玻璃类型的文物内部氧化钙含量存在显著差异，对于高钾玻璃文物内部氧化钙平均含量为 3.8445%，对于铅钡玻璃文物内部氧化钙平均含量为 2.05%，高钾玻璃文物内部氧化钙平均含量显著高于铅钡玻璃，高 1.795%。

对于氧化锶：独立样本 t 检验的 p 值为 $3.1002e-06^{***}$，可认为不同玻璃类型对于文物内部氧化锶含量在 99% 的置信水平下呈现显著相关性，意味着不同玻璃类型的文物内部氧化锶含量存在显著差异，对于高钾玻璃文物内部氧化锶平均含量为 0.0278%，对于铅钡玻璃文物内部氧化锶平均含量为 0.3480%，高钾玻璃文物内部氧化钙平均含量显著低于铅钡玻璃，低 0.3202%。

5.2.8 组间非参数检验

本文使用威尔克森符号秩检验来进行表 9中的非参数检验。

威尔科克森符号秩检验是是一种非参数检验，但在使用 “非参数” 一词时，通常意味着总体数据不服从正态分布，并不意味着对总体一概不知。通过威尔科克森符号秩检验，可以在不假定数据服从正态分布的前提下，判断出相应的数据总体分布是否相同。

针对该问题，对不同的化学成分，本文将 “H_0：有无风化（或不同玻璃类型）的文物样品的化学成分含量是相似的” 作为原假设，相应的 “H_1：有无风化（或不同玻璃类型）的文物样品的化学成分含量是不同的” 作为备择假设，使用威尔科克森符号秩检验进行假设检验。

利用 Python 中的 scipy.stats.ranksums 函数进行编程求解 [4]，本文得到单个化学成分的威尔科克森符号秩检验的 p 值，如下表 10所示。

表 10　单个化学成分的威尔科克森符号秩检验 p 值

化学成分	pvalue	化学成分	pvalue	化学成分	pvalue
氧化钠	0.0446	氧化钾	5.0728e-06	氧化铝	0.0501
氧化铁	0.0219	氧化铜	0.0649	氧化铅	4.4340e-10
氧化钡	8.5622e-09	五氧化二磷	0.2030		

根据上表 10可以发现除氧化铝、氧化铜和五氧化二磷的 p 值大于 0.05 之外，其余化学成分的 p 值都小于 0.05，即对于大部分的化学成分，可以拒绝原假设，即有无风化（或不同玻璃类型）的文物样品的化学成分含量是不同的；对于氧化铝、氧化铜以及五氧化二磷，无法拒绝原假设，即有无风化（或不同玻璃类型）的文物样品的化学成分含量是相似的。

5.2.9 组内比较选择

根据图 6，本文通过观察单个统计图的曲线增长趋势和增长幅度，对上一轮组间比较选择之后的化学成分进行如下的组内比较选择。

表 11　组内比较选择

化学成分	比较选择	正态检验 p 值 (组 1)	正态检验 p 值 (组 2)	分析方法
氧化钠	无风化内部比较高钾铅钡	0.1823	0.0593	独立样本 t 检验
氧化钾	高钾内部比较有无风化	0.2917	0.6955	独立样本 t 检验
氧化钙	高钾内部比较有无风化	0.4793	0.9037	独立样本 t 检验
氧化钙	铅钡内部比较有无风化	0.0414	0.6209	独立样本 t 检验
氧化铝	高钾内部比较有无风化	0.7226	0.7860	独立样本 t 检验
氧化铁	高钾内部比较有无风化	0.2514	0.7791	独立样本 t 检验
氧化铜	高钾内部比较有无风化	0.7222	0.6725	独立样本 t 检验
氧化铅	铅钡内部比较有无风化	0.4396	0.9974	独立样本 t 检验
氧化钡	铅钡内部比较有无风化	0.0001	0.0387	非参数检验
五氧化二磷	高钾内部比较有无风化	0.1042	0.8968	独立样本 t 检验
五氧化二磷	铅钡内部比较有无风化	1.1543e-05	0.4760	非参数检验
氧化锶	铅钡内部比较有无风化	0.0350	0.5161	非参数检验

表 11中根据组内比较选择，将原本的四大部分数据合并成两大部分，分别进行正态分布检验，若组内的两组数据都服从正态分布，使用独立样本 t 检验进行分析，否则使用非参数检验。

5.2.10 组内独立样本 t 检验

对于该部分而言，即为分析在固定表面是否风化或玻璃类型下，有无风化（或玻璃类型为高钾还是铅钡）对文物样品内化学成分（氧化钙和氧化锶）含量的差异情况。

本文使用 Pyhton 中的 ttest_ind 函数对表 11中的独立样本 t 检验进行求解，得到以下结果。

表 12　组内独立样本 t 检验 p 值

化学成分	p 值	化学成分	p 值	化学成分	p 值
氧化纳	0.1903	氧化钾	$6.0014e-05^{***}$	氧化钙（高钾）	0.0032^{***}
氧化钙（铅钡）	0.0024^{***}	氧化铝	0.0005	氧化铁	0.0283
氧化铜	0.2438	氧化铅	7.2506e-09	五氧化二磷	0.0786

对于无风化样本内的氧化钠：可以在 95% 的置信水平下认为玻璃类型与无风化样本内部的氧化纳含量没有显著相关性。

对于高钾玻璃样本：

氧化钾：可认为表面是否风化对于文物内部氧化钾的含量在 99% 的置信水平下呈现显著相关关系，意味着高钾玻璃文物中表面风化文物与表面风化文物内部氧化钾含量存在显著差异，对于表面无风化文物内部的氧化钾平均含量为 9.3308%，对于表面有风化文物内部的氧化钾平均含量为 0.5433%，无风化文物内部氧化钾平均含量显著高于有风化文物，高 8.7875%。

氧化钙：可认为表面是否风化对于文物内部氧化钙的含量在 99% 的置信水平下呈现显著相关关系，意味着高钾玻璃文物中表面风化文物与表面风化文物内部氧化钙含量存在显著差异，对于表面无风化文物内部的氧化钙平均含量为 5.3325%，对于表面有风化文物内部的氧化钙平均含量为 0.87%，无风化文物内部氧化钙平均含量显著高于有风化文物，高 4.4625%。

氧化铝：可认为表面是否风化对于文物内部氧化铝的含量在 99% 的置信水平下呈现显著相关关系，意味着铅钡玻璃文物中表面风化文物与表面风化文物内部氧化铝含量存在显著差异，对于表面无风化文物内部的氧化铝平均含量为 6.62%，对于表面有风化

文物内部的氧化铝平均含量为 1.93%，无风化文物内部氧化铝平均含量显著低于有风化文物，低 4.69%。

氧化铁：可认为表面是否风化对于文物内部氧化铁的含量在 95% 的置信水平下呈现显著相关关系，意味着铅钡玻璃文物中表面风化文物与表面风化文物内部氧化铁含量存在显著差异，对于表面无风化文物内部的氧化铁平均含量为 1.9317%，对于表面有风化文物内部的氧化铁平均含量为 0.265%，无风化文物内部氧化铝平均含量显著高于有风化文物，高 1.6667%。

氧化铜：可以在 95% 的置信水平下认为是否风化与高钾玻璃样本内部的氧化铜含量没有显著相关性。

五氧化二磷：可以在 95% 的置信水平下认为是否风化与高钾玻璃样本内部的五氧化二磷含量没有显著相关性。

对于铅钡玻璃样本：

氧化钙：可认为表面是否风化对于文物内部氧化钙的含量在 99% 的置信水平下呈现显著相关关系，意味着铅钡玻璃文物中表面风化文物与表面风化文物内部氧化钙含量存在显著差异，对于表面无风化文物内部的氧化钙平均含量为 1.3204%，对于表面有风化文物内部的氧化钙平均含量为 2.6954%，无风化文物内部氧化钙平均含量显著低于有风化文物，低 1.3749%。

氧化铅：可认为表面是否风化对于文物内部氧化铅的含量在 99% 的置信水平下呈现显著相关关系，意味着铅钡玻璃文物中表面风化文物与表面风化文物内部氧化铅含量存在显著差异，对于表面无风化文物内部的氧化铅平均含量为 22.0848%，对于表面有风化文物内部的氧化钙平均含量为 43.3138%，无风化文物内部氧化钙平均含量显著低于有风化文物，低 21.2291%。

5.2.11 组内非参数检验

继续使用威尔克森符号秩检验来进行表 11中的非参数检验。

针对该问题，对不同的化学成分，本文将 “H_0：确定玻璃类型下是否风化的文物样品的化学成分含量是相似的” 作为原假设，相应的 “H_1：确定玻璃类型下是否风化的文物样品的化学成分含量是不同的” 作为备择假设，使用威尔科克森符号秩检验进行假设检验。

利用 Python 中的 scipy.stats.ranksums 函数进行编程求解，本文得到单个化学成分的威尔科克森符号秩检验的 p 值，如下表 13所示。

表 13　单个化学成分的威尔科克森符号秩检验 p 值

化学成分	pvalue	化学成分	pvalue	化学成分	pvalue
氧化钡	0.5215	五氧化二磷	0.0001	氧化锶	0.0313

根据上表 13可以发现除氧化钡的 p 值大于 0.05 之外，其余化学成分的 p 值都小于 0.05。对于五氧化二磷和氧化锶，可以拒绝原假设，即确定玻璃类型下是否风化的文物样品的五氧化二磷和氧化锶含量是不同的；对于氧化钡无法拒绝原假设，即确定玻璃类型下是否风化的文物样品的氧化钡含量是相似的。

5.2.12 单独分析

根据图 6可知，对于氧化锡：只有少量铅钡玻璃表面风化样本内部存在 0.5%~1.5% 之间的氧化锡，以及一个无风化高钾玻璃样本内部存在超过 2.0%，本文认为这些是少数特例，可能跟样本本身有关，与样本表面有无风化或样本玻璃类型没有显著相关关

系；对于二氧化硫：无论风化与否还是玻璃类型，绝大多数样本内部的二氧化硫含量都为 0%，本文认为表面风化的铅钡玻璃样本中少部分存在较高二氧化硫含量这一现象可能跟这些文物埋藏的特俗环境相关 [5]，与样本表面有无风化或玻璃类型没有显著相关关系。

5.3 问题一：化学成分预测

5.3.1 思路分析

根据问题分析，本文给出思路分析图 7。

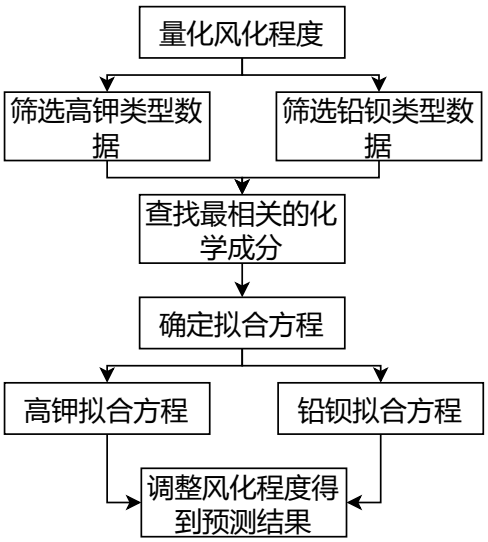


图 7　问题一第三小问思路分析图

5.3.2 量化风化程度

由于风化程度相比于是否风化，需要考虑“程度”这一因素，因而需要对风化程度进行量化。

对于风化程度，存在未风化、风化和严重风化三种，这三种存在风化程度上的递进关系，并且风化和未风化之间的风化程度差距与严重风化和风化之间的差距不能简单的认为是一样的，故本文出于对这一层面的考虑，将风化程度确定为由虚拟变量 t_1, t_2 共同决定的变量。通过利用两个虚拟变量对风化程度进行进行以下刻画：

$$\text{风化程度} = p_1 t_1 + p_2 t_2 \tag{8}$$

其中，p_1, p_2 为需要拟合的参数。这样可以通过两个拟合参数分别衡量风化和未风化之间的风化程度差距以及严重风化和风化之间的风化程度差距。

对于未风化的文物样本，设定其风化程度的两个虚拟变量为 $t_1 = 0, t_2 = 0$，对于风化文物样本，设定其风化程度为 $t_1 = 1, t_2 = 0$，对于严重风化的文物样本，设定其风化程度为 $t_1 = 0, t_2 = 1$。即用 t_1 衡量是否风化，用 t_2 衡量是否严重风化。

5.3.3 筛选数据

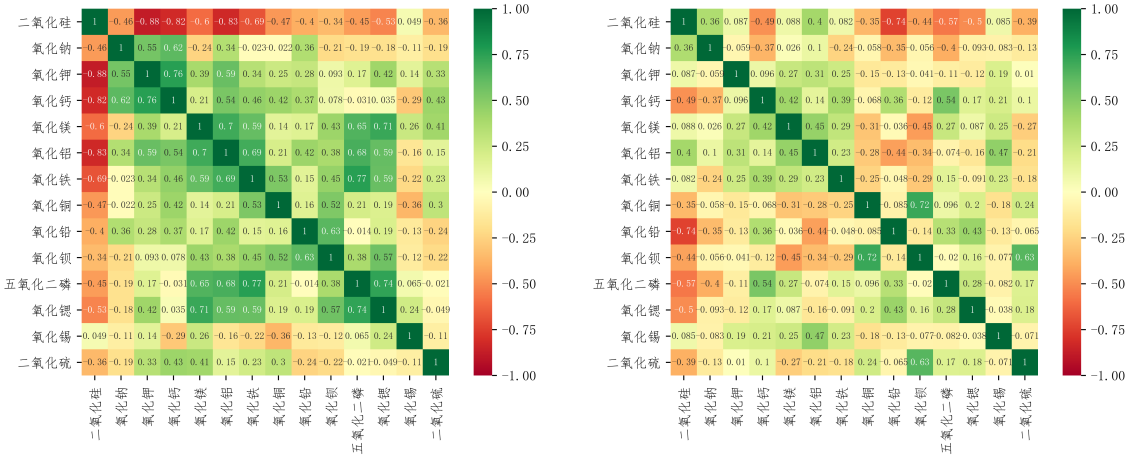
只需将不同玻璃类型的有效数据分别筛选出来作为新表即可。

5.3.4 查找最相关的化学成分

本文考虑到矿物中化学成分之间往往存在的共生现象 [6]，因而需要将除需要预测的化学成分之外的所有化学成分纳入到自变量当中，但是有部分的化学成分与当前需要预测的化学成分之间没有相关关系，这样会导致预测的结果受不相关的、随机的化学成分变化影响程度较大，实际的预测效果会很差。于是本文在综合考虑上述因素后，先对需要预测的化学成分与其他化学成分之间的相关关系进行分析，找出最相关的 3 个化学成分，将这 3 个化学成分作为变量加入到预测函数当中进行拟合。

针对需要预测的化学成分（以下称目标化学成分）与其余化学成分之间的相关关系，本文求解出目标化学成分与其余化学成分之间的 Pearson 相关系数，从中挑选相关系数最大的 3 个化学成分，作为该目标化学成分的的预测函数的 3 个变量 x_1, x_2, x_3。而由于不同玻璃类型之间的化学成分的相关关系可能存在差异，本文将两种玻璃类型单独讨论，分别求解相关系数矩阵。

本文按照上述的思路进行 Pearson 相关系数的求解，得到以下热图 8。



(a) 高钾类型　　(b) 铅钡类型

图 8　化学成分 Pearson 相关系数矩阵

根据上图 8，从矩阵每行当中取出颜色最深、绝对值最大的 3 个相关系数（对角线除外），将其对应的 3 个化学成分记录下来作为后续该目标化学成分的预测方程的 3 个自变量。

5.3.5 确定拟合方程

本文对不同玻璃类型拟合不同的预测方程，对于某一指定的目标化学成分（例如二氧化硅），分别用表 4（高钾类型）和表 5（铅钡类型）的数据进行拟合，其拟合方程如下：

$$\begin{aligned} y =& p_{11}x_1^2 + p_{22}x_2^2 + p_{33}x_3^2 + p_{12}x_1x_2 + p_{13}x_1x_3 + p_{23}x_2x_3 \\ &+ p_1x_1 + p_2x_2 + p_3x_3 + p_{t1}t_1 + p_{t2}t_2 + p_0 \end{aligned} \tag{9}$$

其中 p 为需要拟合的参数，x_1, x_2, x_3 为前文提到的 3 个与目标化学成分对相关的化学成分的含量，t_1, t_2 为风化程度的两个虚拟变量，y 为目标化学成分的含量。

分别拟合得到二氧化硅的两个方程，见式 (10) 和式 (11)，其中式 (10) 为高钾类型的预测方程，式 (11) 为铅钡类型的预测方程。两个方程的拟合优度 R 分别为：0.9954（高钾）和 0.8762（铅钡），对于其他化学成分的回归拟合优度，绝大部分都大于 0.8，说明拟合效果较好。

$$\begin{aligned} y =& -0.07x_1^2 + 0.16x_2^2 - 0.31x_3^2 + 0.03x_1x_2 - 0.16x_1x_3 + 0.14x_2x_3 \\ &+ 1.07x_1 - 3.99x_2 + 3.67x_3 + 18t_1 + 115.4t_2 + 73.85 \end{aligned} \tag{10}$$

$$\begin{aligned} y =& 0.008x_1^2 + 0.04x_2^2 - 11.5x_3^2 + 0.07x_1x_2 - 0.05x_1x_3 + 2.4x_2x_3 \\ &- 1.16x_1 - 4.64x_2 - 8.57x_3 - 12.04t_1 - 34.4t_2 + 82.58 \end{aligned} \tag{11}$$

5.3.6 得到预测结果

根据已经获得的预测方程，本文通过调整 t_1, t_2 的值（$t_1 = 0, t_2 = 0$），再代入目标化学成分的相关自变量，即可根据上面的两个方程式 (10) 和式 (11) 得到高钾类型的文物中该目标化学成分未风化前的含量和铅钡类型的文物中该目标化学成分未风化前的含量。

以下展示部分风化采样点的部分化学成分含量预测，见表 14。

表 14　部分预测结果

文物采样点	二氧化硅	氧化钠	氧化钾	氧化钙	氧化镁	氧化铝
07	76.11	1.33	6.44	5.41	0.93	3.36
09	76.35	1.65	7.80	4.82	1.02	3.15
10	76.62	1.99	8.70	4.64	1.03	2.59
12	76.70	1.43	7.31	4.30	1.12	2.89
22	77.92	0.98	6.79	4.51	1.33	3.11

同时为了检验预测结果的正确性，本文对各个需要预测的风化样本点的每个化学成分的预测值进行累加，得到以下结果见图 9。其中可以看出，虽然存在少数预测数据累加和不在有效数据范围内，但是大部分预测数据累加和满足有效数据条件，而且由于本文所提出的模型没有加上有效数据的限制，可以看出在不限制预测范围的情况下，该模型仍然可以获得较为合理和正确的预测数据，体现该模型具有一定的合理性。

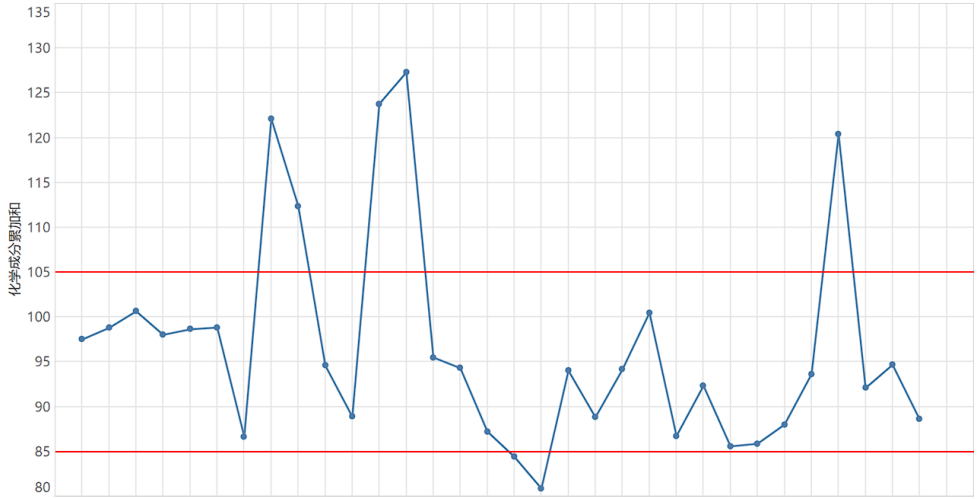


图 9　预测值累加结果

5.4 问题二：分类规律分析

根据问题分析，使用逻辑回归来对分类规律进行拟合，以得到一个显性的表达式。

本文将玻璃类型作为一个二分类变量，使用逻辑回归，利用题目给出的数据，通过极大似然估计得到参数的估计，即可得到分类规律的线性表示。

逻辑回归实际上就是将连续函数 $F(x,\beta)$ 设置为逻辑分布的累计分布函数，即：

$$\hat{y} = F(x,\beta) = \frac{\exp\left(\sum_{i=1}^{16} x_i\beta_i\right)}{1+\exp\left(\sum_{i=1}^{16} x_i\beta_i\right)} \tag{12}$$

其中，x_i 为输入的变量（14 个化学成分，1 个表面是否风化，一个常数项 x_{16} 恒为 1），β_i 为对应的拟合系数，$\hat{y}$ 为累计分布函数值。

由于 $\hat{y}$ 的值位于 $[0,1]$ 区间当中，根据 $\hat{y}$ 的值的大小，小于 0.5 的数据归为 0（高钾）类，大于 0.5 的数据归为 1（铅钡）类，等于 0.5 的情况几乎不可能发生，不予讨论。

使用极大似然估计对式 (12) 中的参数进行拟合，具体操作使用 Python 中的 LogisticRegression 函数进行参数的拟合，其参数设置为 “max_iter=1000”。最后得到以下参数

具体值列于表 15，其中 β_1 为有无风化，β_2~β_{15} 为附表二中的化学成分（按顺序），β_{16} 为常数项（对应 $x_{16}=1$）。

表 15　逻辑回归参数拟合结果

β_1	β_2	β_3	β_4	β_5	β_6	β_7	β_8
-0.0043	-0.136	0.015	-0.2734	-0.0811	-0.0028	0.0736	-0.0735
β_9	β_{10}	β_{11}	β_{12}	β_{13}	β_{14}	β_{15}	β_{16}
-0.0483	0.4734	0.1315	-0.0646	0.0054	0.0005	0.0012	5.8096

根据式 (12)，将系数代入，即可得到对文物玻璃类型的分类规律。将不同文物的数据代入自变量 x_i 的相应位置，即可得到对该文物的玻璃类型划分。

5.5 问题二：亚类划分

5.5.1 思路分析

根据问题分析，本文给出以下思路分析图 10。

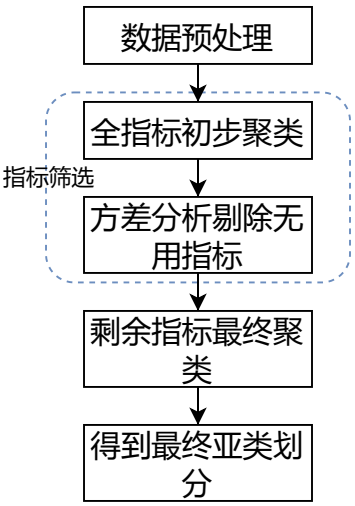


图 10　问题二第二小问思路分析图

5.5.2 数据预处理

根据问题分析，将附件表二中的有效数据中的风化样本数据代入问题一得到的目标化学成分预测模型，得到对应的目标化学成分风化前的预测值。将所有风化前化学成分含量预测值与原表中未风化数据整合，得到新表 5，即可完成数据预处理。

经过该处理，可以将有无风化对接下来的聚类的影响降低到最小，且符合控制变量的思想，本文认为这一处理是较为合理的。

5.5.3 指标筛选

本文使用聚类来完成亚类划分，但是对于具体使用的聚类方法，目前较为普遍的方法有 K-Means 算法或者系统聚类。本文选择系统聚类作为接下来进行聚类的方法。

为了更好的得到精确合理的亚类划分，本文将所有的化学成分纳入考虑，进行初步的系统聚类，得到初步的亚类划分，并将每一个化学成分与划分得到的亚类集合（分类变量）进行方差分析，确定对应的化学成分对于亚类的划分是否存在显著影响，将没有显著影响的化学成分剔除，剩下的即为本文认为的“合适的”化学成分。

a. 系统聚类

对于系统聚类，其算法流程可概述成以下伪代码：

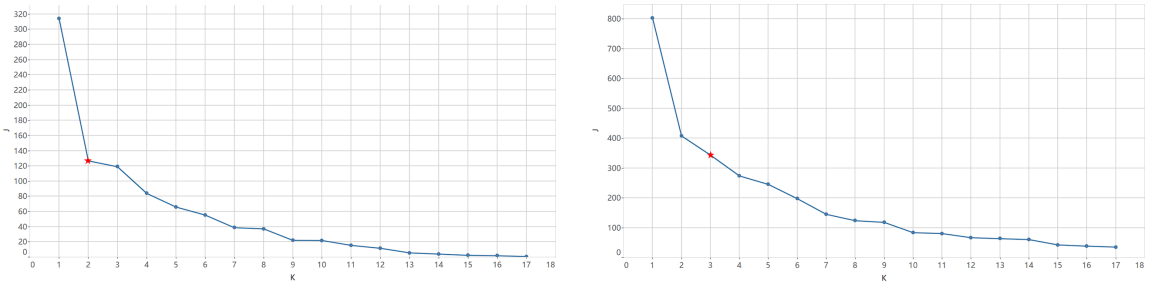
为了更加科学的确定最后的亚类数目，本文接下来使用肘部法则确定最优的聚类数量，即亚类的数目。

为此先简单介绍肘部法则。通过定义各个类的畸变程度与所有类的总畸变程度，绘制聚合系数折线图，选取总畸变程度显著降低的点的横坐标作为最优聚类数目。

对于各类畸变程度的定义，本文采用将该类重心与其内部成员位置距离的平方和作为各类的畸变程度定义。对于某一个类 C，其畸变程度为：$\sum_{x_i\in C}|x_i-u_i|^2$。所有类的总畸变程度为：$J=\sum_{k=1}^{K}\sum_{x_i\in C}|x_i-u_i|^2$，其中 K 为类别个数。以 (K,J) 为坐标点格式，绘制 K 的数量变化与 J 的改变关系，即为聚合系数折线图。

聚类算法
1、将新表 5 中每一个取样点看作一类，计算取样点两两之间的最小距离（同一化学成分的欧氏距离最小值） 2、将距离最小的两个取样点合并成一个新类 3、重新计算新类与其他新类的距离 4、重复 2，3 步直到所有类最后合并成一类 5、结束

通过上述操作，本文最终绘制出以下肘部图 11。



(a) 高钾　　(b) 铅钡

图 11　肘部图

根据图 11，依照肘部法则的判断标准，确定的亚类数目为：高钾玻璃共有 2 个亚类，铅钡玻璃共有 3 个亚类。

b. 方差分析

在进行完系统聚类之后，本文对已知的高钾玻璃的 2 个亚类和铅钡玻璃的 3 个亚类分别与单个化学成分进行方差分析。将高钾玻璃的 2 个亚类作为一个分类变量，与单个化学成分进行方差分析，分析可以得到系统聚类产生的 2 个亚类之间该化学成分是否存在显著差异。其原假设是没有显著差异，备择假设是有显著差异。由于前文已经叙述过方差分析的过程，这里不再赘述，直接给出部分分析结果，详细结果见附录。

表 16　部分方差分析结果

观测变量	控制变量	df	sum_sq	mean_sq	F	PR(>F)
二氧化硅	高钾亚类	1.0	969.530328	969.530328	79.084197	1.369100e-07
氧化纳	高钾亚类	1.0	0.010617	0.010617	0.008037	0.929679
氧化钾	高钾亚类	1.0	89.302826	89.302826	13.567066	0.002012
氧化钙	高钾亚类	1.0	25.590261	25.590261	5.015565	0.039675
二氧化硅	铅钡亚类	1.0	3773.311427	3773.311427	55.071027	1.885339e-09
氧化钠	铅钡亚类	1.0	8.860923	8.860923	3.208402	0.079702
氧化钾	铅钡亚类	1.0	0.035833	0.035833	0.61344	0.437424
氧化钙	铅钡亚类	1.0	1.098871	1.098871	0.813509	0.371684

根据表 16以及附表，可以知道对于高钾亚类而言，二氧化硅、氧化钾、氧化钙、氧化铝、氧化铜和氧化锡六个化学成分在不同亚类之间含量存在显著差异；对于铅钡亚类而言，二氧化硅、氧化铅、五氧化二磷和氧化锶四个化学成分在不同亚类之间含量存在显著差异。

由此可知，对于高钾亚类而言，选出 6 个对系统聚类的亚类划分产生影响的化学成分；对于铅钡亚类而言，选出 4 个对系统聚类的亚类划分产生影响的化学成分。接下来就利用选出的这些化学成分，进行最后的系统聚类。

5.5.4 最终聚类

在前面筛选出 “合适的” 化学成分的基础之上，本文利用这些化学成分，对高钾玻璃和铅钡玻璃分别进行最后的系统聚类，只用筛选后的化学成分。由于前文已经介绍过系统聚类的方法，在此不做赘述，直接给出最终的聚类结果：高钾玻璃：共 2 个亚类；铅钡玻璃：共 3 个亚类；具体划分结果详情见附表，此处展示部分划分结果见表 17。

表 17　部分聚类结果

文物采样点	玻璃类型	亚类	文物采样点	玻璃类型	亚类
07	高钾	1	55	铅钡	1
09	高钾	1	54 严重风化点	铅钡	2
01	高钾	2	30 部位 1	铅钡	2
03 部位 2	高钾	2	48	铅钡	3
50 未风化点	铅钡	1	28 未风化点	铅钡	3

5.6 问题二：合理性与敏感性分析

5.6.1 合理性分析

a. 数值层面

本文使用轮廓系数对系统聚类得到的结果进行数值上的分析，以下先简单介绍轮廓系数的计算方法。

针对该题，将每一个样本点的筛选后的化学成分数据组合成一个向量 v，对于每一个亚类中的每个向量，可以通过以下公式得到其单独的轮廓系数：

$$S(v)=\frac{b(v)-a(v)}{\max\{a(v),b(v)\}} \tag{13}$$

其中，$a(v)$ 为 v 向量到同一亚类中其他不同向量的距离的平均值；$b(v)$ 为 v 向量到其他亚类的某一向量的距离的最小值。由此看出轮廓系数介于 $[-1,1]$ 之间，且该系数越大，证明分类的效果越好，分类越合理。

对于整个聚类结果的轮廓系数，只需将单个类别的轮廓系数取平均即可得到。

本文通过使用 Python 中的 metrics.silhouette_score 函数，分别计算两种玻璃类型使用系统聚类的轮廓系数，得到以下结果：高钾玻璃轮廓系数为 0.5932，铅钡玻璃轮廓系数为 0.3874，可见本文使用的聚类模型具有较好的效果，合理性较好。

b. 实际生活层面

本文通过查阅文献 [7]，本文获得以下玻璃内部化学成分与外部体现出来的理化特性之间的关系，见表 18。

另外，二氧化锡加入玻璃通常作为着色剂，乳化剂，会使玻璃透光性下降；氧化铜加入玻璃会使玻璃呈现出蓝红绿等不同的颜色，使玻璃颜色更加鲜艳；而五氧化二磷用于制造光学玻璃、透紫外线玻璃、隔热玻璃、微晶玻璃和乳浊玻璃等，以提高玻璃的色散系数和透过紫外线的能力。

本文根据上述化学成分与理化性质之间的关系，对不同亚类之间上述化学成分变化所带来的亚类本身的理化特性进行整合，得到以下结果：

表 18　化学成分与理化特性的关系

化学成分	相对密度	热稳定性	机械强度	光泽	硬度
二氧化硅	+	+	+		
氧化铅	+			+	
氧化铝			+		
氧化钾		−		+	
氧化钙		−	+		+
氧化锶				+	

高钾玻璃：

亚类 1：透光性差、密度小、热稳定性好、机械强度大；

亚类 2：机械强度大、热稳定性差、有光泽、硬度大、颜色鲜艳；

铅钡玻璃：

亚类 1：透光性强、隔热性强；

亚类 2：密度大、有光泽、折射率高；

亚类 3：密度小、热稳定性差、机械强度大；

由此可以看出，不同亚类确实可以在实际生活当中通过理化特性上的区别得以区分，进一步证明本文得到的亚类划分具有一定的现实意义，合理性较好。

5.6.2 敏感性分析

a. 思路分析

根据问题分析，本文给出以下敏感性分析的流程图 12。

b. 锚点确定

由于前文在建立聚类模型时对化学成分进行了筛选，本文接下来只针对选定的化学成分进行敏感性分析。对于需要进行分析的单个化学成分，依旧称其为目标目标化学成分。

本文对不同玻璃类型的单个亚类的目标化学成分的平均值 $\hat{Q}$ 进行计算，并选定 1 或 3 个目标化学成分最接近平均值的样本点作为该亚类的该目标化学成分的“锚点”。对于锚点个数的确定，需要根据该亚类所含样本点个数进行判断，如果该亚类中的样本点个数小于 5，则只选取 1 个锚点；反之，选取 3 个锚点。

c. 重新聚类

选取一个该亚类中除锚点以外的样本点，对其中的目标化学成分的含量进行调整。为了模拟可能出现的化学成分含量测量时的误差，本文选择将目标化学成分的含量上下分别调整 5%，10%、20%。

对于样本点目标化学成分含量的不同改变，分别重新进行前文的亚类划分，得到新的划分结果。

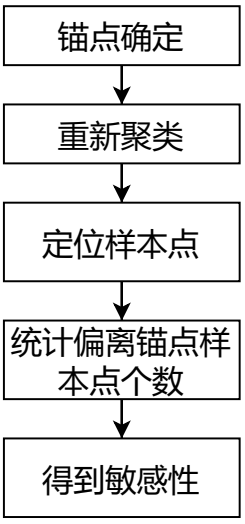


图 12　敏感性分析流程图

d. 定位样本点

对于选定的样本点，需要在新的划分结果中定位选择的锚点以及该样本点。由于锚点是通过距离平均值最近来进行选择，根据前文介绍的系统聚类的原理可知，在其他化学成分不变的情况下，对于某一给定的化学成分，最接近类内平均值的样本点应当在该类的聚类中心附近，因而在只改变一个样本点的目标化学成分含量时不会影响锚点的聚类分类。从而可将锚点所在类标记为该样本点应当所属类，与样本点实际所属类进行对比，若不为同一类，则记录为 1；反之，记录为 0。

e. 统计偏离锚点样本点个数

根据前文的记录，对于每一个亚类，不同目标化学成分，不同调整比例，不同样本点，都可以得到一个记录。对给定的亚类，指定目标化学成分和调整比例，将所有记录相加，得到偏离锚点样本点个数 n。

f. 衡量敏感性

本文为了衡量敏感性，设定指标抗干扰能力 η 的计算方式：

$$\eta = 1 - \frac{n}{N} \tag{14}$$

其中，n 为偏离锚点样本点个数，N 为给定亚类和目标化学成分下的非锚点样本点个数。

根据前面对偏离锚点样本点的定义不难看出，η 可以很好的衡量在一定的调整比例下，目标化学成分对于样本点数据波动的抗干扰能力。

本文利用 Python 编程实现 η 的计算，给出以下结果，铅钡玻璃不同目标化学成分的抗干扰能力见表 19，高钾玻璃不同目标化学成分的抗干扰能力见表 20。

表 19　铅钡玻璃不同目标化学成分的 η 值

调整比例	二氧化硅	氧化铅	五氧化二磷	氧化锶
5%	0.952380952	0.976190476	1	1
-5%	0.976190476	1	1	1
10%	0.880952381	0.928571429	1	1
-10%	0.904761905	1	1	1
20%	0.761904762	0.880952381	1	1
-20%	0.80952381	0.928571429	1	1

表 20　高钾玻璃不同目标化学成分的 η 值

调整比例	二氧化硅	氧化钾	氧化钙	氧化铝	氧化铜	氧化锡
5%	0.833333	1	1	1	1	1
-5%	1	1	1	1	1	1
10%	0.75	1	1	1	1	1
-10%	0.416667	1	1	1	1	1
20%	0.166667	1	1	1	1	1
-20%	0.166667	1	1	1	1	1

由两表可以看出，不论玻璃类型，二氧化硅对于数据波动的抗干扰能力都是所有指标里面最差的，可见二氧化硅对模型输入的数据敏感；对于高钾玻璃而言，除了二氧化硅以外的化学成分对输入数据都不敏感；对于铅钡玻璃而言，除了二氧化硅之外，氧化铅对输入数据也具有一定的敏感性，但敏感性较弱。可见本文得到的模型对于输入数据的敏感性较弱，较为稳定，可以较好的对不同玻璃类型的亚类进行划分。

5.7 问题三：玻璃类别分类

根据问题分析，使用问题二中的逻辑回归模型式 (12)，以及拟合出来的参数表 15，将附件表三的待分类采样点数据代入式 (12) 当中，即可得到一个介于 $[-1, 1]$ 之间的值，根据值的大小，小于 0.5 则将该文物归为 0（高钾）类，大于 0.5 这归为 1（铅钡）类，对于等于 0.5 的极端特殊情况，由于实际求解得到的值不存在 0.5，本文不予考虑。

通过调用 Python 中的 LogisticRegression 函数，对未知类型的文物数据进行分类求解，得到以下分类结果，见表 21。

表 21　未知文物类型分类

文物编号	分类结果	分类正确概率	文物编号	分类结果	分类正确概率
A1	0(高钾)	0.99428	A5	1(铅钡)	0.97662
A2	1(铅钡)	1	A6	0(高钾)	0.99932
A3	1(铅钡)	1	A7	0(高钾)	0.99867
A4	1(铅钡)	1	A8	1(铅钡)	0.99995

5.8 问题三：敏感性分析

5.8.1 模型本身的敏感性

本文使用的逻辑回归模型在实际训练当中需要对训练的参数进行设置，由于不同参数设置可能导致模型的训练结果存在偏差，对应得到的拟合参数存在偏差，使得最后的分类结果不同。出于对此层面的考虑，本文对模型的训练参数进行调整，以分析模型对于本身参数设置的敏感性。

针对该问题，本文对训练时的 “tol” 和 “c” 两个参数进行单独调整，使用问题二中的数据集，将该数据集随机分成 7 份和 3 份，利用 7 份数据进行参数调整后的式 (12) 的参数拟合，利用剩下 3 份数据进行验证分析。

对于 “tol” 和 “c” 两个参数的调整，本文采用在规定区间 $[0.000001, 100]$ 中等间距选取 300 个数据点，对一次测试只调整一个参数，满足控制变量的原则。

对于参数的拟合，与前文相同，只是使用 Python 中的 LogisticRegression 函数时，需要根据当前的参数训练参数调整需要，进行调整。

对于最后的验证分析，本文采用将 3 份验证数据使用拟合得到的新的参数和模型进行分类的求解，得到相应的预测为 01 类型的概率分数。对于同一个采样点，以原来的分类数据为真实分类标准，从对应采样点的概率分数中挑选出对应类别的概率分数。对 3 份数据当中的每一个采样点都进行以上操作，得到概率分数向量。

经过上述过程，对于每个训练参数调整后的模型分类结果，可以得到概率分数向量 v，以及原始 3 份数据的分类向量 V。对与这两个向量，求解他们之间的 Logloss 函数值，以此作为衡量分类效果的指标。

Logloss 损失函数常用于衡量逻辑回归模型的分类效果，其函数值的计算方法如下：

$$Logloss = -\left(y \log\left(p\right) + \left(1 - y\right) \log\left(1 - p\right)\right) \tag{15}$$

其中，y 为 3 份数据的真实分类标签，p 为模型预测为正样本的概率。

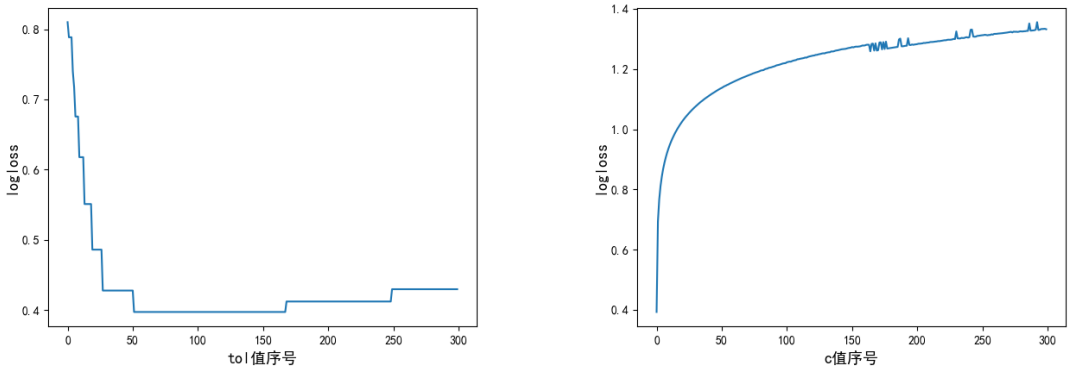
通过上述过程可以得到每个训练参数取值下的 Logloss 函数值，据此可以分析该模型对训练参数的敏感性，本文先给出两个参数的不同取值与 Logloss 函数值的曲线图 13。

根据图 13可知，Logloss 函数值随训练参数改变会呈现迅速上升（或迅速下降）然后平缓的现象，因此在训练参数较小时，训练参数的微小改变会对模型的分类造成较大影响，于是本文认为该模型对训练参数的改变的敏感性较强，在实际分类时需要对参数的选择上进行进一步考究。

5.8.2 模型对输入数据的敏感性

该模型一共有 14 个化学成分和 1 个表面风化共 15 个输入变量，对于 14 个化学成分，本文对其输入的波动对模型分类结果的影响进行分析，以反映模型对输入数据的敏感性；对于表面风化这一分类变量，由于更改这一变量应该导致对应的化学成分发生变化，本文不将其纳入考虑。

首先，对输入的单个化学成分数据进行调整，得到调整后的分类结果（预测为 01 类型的概率分数）。再对于同一个采样点，以上一问对未知类别文物的玻璃类别分类结



(a) 参数 tol 变化 **(b) 参数 c 变化**

图 13　Logloss 函数值变化情况

果为标准，从对应采样点的概率分数中挑选出对应类别的概率分数。对附件表三当中的每一个采样点都进行以上操作，得到概率分数向量 v。

接下来同样进行 Logloss 值的计算，根据式 (15)，求解概率分数向量 v 和分类向量 V 的 Logloss 函数值，得到以下结果，见图 14。

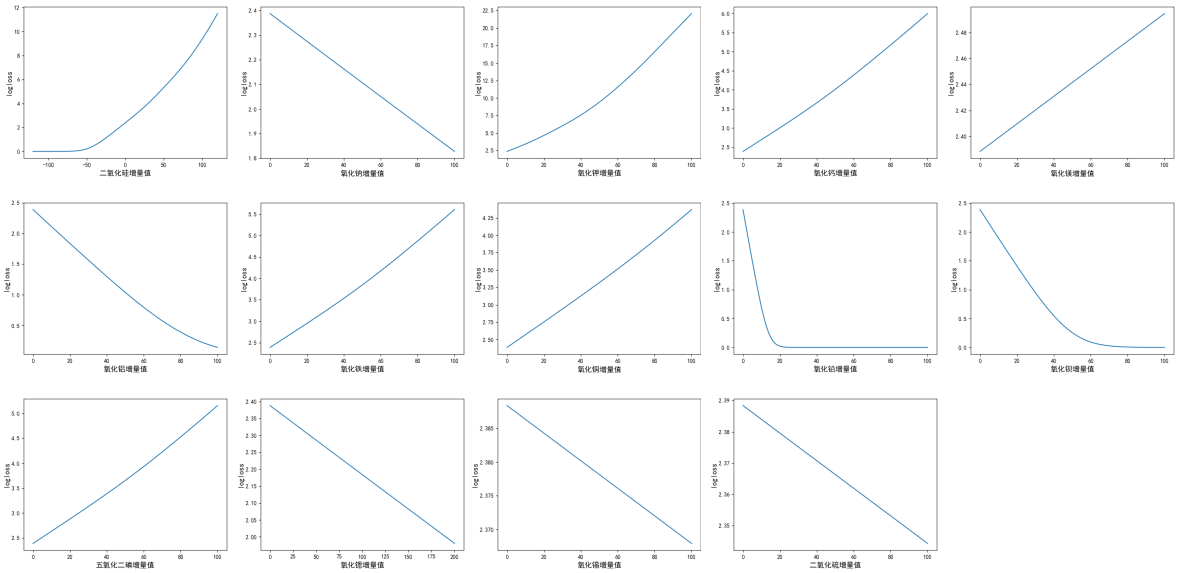


图 14　Logloss 函数值变化情况

根据图 14，可以发现大部分的 Logloss 函数值曲线呈现类似一次函数的形式，证明在改变少量化学成分含量的情况下，模型的分类不会出现显著改变。对于二氧化硅，由于所有样本的二氧化硅含量都较大，本文设置减少一定含量的数据输入情况，从图中可以看出：在二氧化硅较少的情况下，其含量变化影响较大，后续的影响较小，但是对于实际情况而言，玻璃样本中的二氧化硅含量一般不会低于 50%，于是可以认为在实际的分类当中二氧化硅的含量对模型的分类结果影响较小。对于氧化铅和氧化钡，由于这两个化学成分作为铅钡玻璃的主要划分依据，在其含量较大时会直接导致类型判定是确定的，于是在两者含量较大时，对模型的分类结果影响较小。

总体上看，该模型对于输入数据的敏感性较弱，模型的稳定性较好。

5.9 问题四：关联关系分析

根据问题分析，本文直接使用前文获得的 Pearson 相关系数矩阵，见图 8，利用其中的不同化学成分之间的 Pearson 相关系数，进行 Pearson 相关系数检验，得到以下检验结果。其中 图 15a为高钾玻璃的检验结果，图 15b为铅钡玻璃的检验结果。

	二氧化硅	氧化钠	氧化钾	氧化钙	氧化镁	氧化铝	氧化铁	氧化铜	氧化铅	氧化钡	五氧化二磷	氧化锶	氧化锡	二氧化硫
二氧化硅	0.000***	0.056*	0.000***	0.000***	0.008***	0.000***	0.002***	0.052*	0.102	0.161	0.062*	0.025**	0.848	0.146
氧化钠	0.056*	0.000***	0.017**	0.006***	0.328	0.161	0.928	0.93	0.143	0.395	0.46	0.47	0.675	0.44
氧化钾	0.000***	0.017**	0.000***	0.000***	0.113	0.010***	0.173	0.316	0.269	0.715	0.497	0.082*	0.574	0.178
氧化钙	0.000***	0.006***	0.000***	0.000***	0.414	0.022**	0.057*	0.080*	0.133	0.757	0.902	0.891	0.243	0.076*
氧化镁	0.008***	0.328	0.113	0.414	0.000***	0.001***	0.009***	0.578	0.509	0.077*	0.003***	0.001***	0.295	0.093*
氧化铝	0.000***	0.161	0.010***	0.022**	0.001***	0.000***	0.002***	0.404	0.081*	0.119	0.002***	0.009***	0.519	0.561
氧化铁	0.002***	0.928	0.173	0.057*	0.009***	0.002***	0.000***	0.025**	0.55	0.060*	0.000***	0.010***	0.382	0.353
氧化铜	0.052*	0.93	0.316	0.080*	0.578	0.404	0.025**	0.000***	0.514	0.027**	0.395	0.458	0.142	0.224
氧化铅	0.102	0.143	0.269	0.133	0.509	0.081*	0.55	0.514	0.000***	0.005***	0.956	0.454	0.598	0.33
氧化钡	0.161	0.395	0.715	0.757	0.077*	0.119	0.060*	0.027**	0.005***	0.000***	0.122	0.015**	0.64	0.388
五氧化二磷	0.062*	0.46	0.497	0.902	0.003***	0.002***	0.000***	0.395	0.956	0.122	0.000***	0.000***	0.799	0.934
氧化锶	0.025**	0.47	0.082*	0.891	0.001***	0.009***	0.010***	0.458	0.454	0.015**	0.000***	0.000***	0.337	0.848
氧化锡	0.848	0.675	0.574	0.243	0.295	0.519	0.382	0.142	0.598	0.64	0.799	0.337	0.000***	0.671
二氧化硫	0.146	0.44	0.178	0.076*	0.093*	0.561	0.353	0.224	0.33	0.388	0.934	0.848	0.671	0.000***

***: p<0.01 **: p<0.05 *: p<0.1

(a) 高钾玻璃检验结果

	二氧化硅	氧化钠	氧化钾	氧化钙	氧化镁	氧化铝	氧化铁	氧化铜	氧化铅	氧化钡	五氧化二磷	氧化锶	氧化锡	二氧化硫
二氧化硅	0.000***	0.011**	0.553	0.000***	0.549	0.004***	0.574	0.012**	0.000***	0.002***	0.000***	0.000***	0.560	0.006***
氧化钠	0.011**	0.000***	0.689	0.006***	0.861	0.483	0.091*	0.692	0.014**	0.700	0.005***	0.526	0.573	0.374
氧化钾	0.553	0.689	0.000***	0.510	0.061*	0.033**	0.086*	0.316	0.370	0.779	0.470	0.407	0.180	0.944
氧化钙	0.000***	0.008***	0.510	0.000***	0.003***	0.354	0.006***	0.642	0.012**	0.395	0.000***	0.240	0.148	0.478
氧化镁	0.549	0.861	0.061*	0.003***	0.000***	0.001***	0.041**	0.031**	0.805	0.001***	0.058	0.554	0.089*	0.064*
氧化铝	0.004***	0.483	0.033**	0.354	0.001***	0.000***	0.112	0.050**	0.002***	0.017**	0.612	0.287	0.001***	0.155
氧化铁	0.574	0.091*	0.086*	0.006***	0.041**	0.112	0.000***	0.087*	0.745	0.040**	0.295	0.535	0.119	0.216
氧化铜	0.012**	0.692	0.316	0.642	0.031**	0.050**	0.087*	0.000***	0.562	0.000***	0.510	0.177	0.207	0.096*
氧化铅	0.000***	0.014**	0.370	0.012**	0.805	0.002***	0.745	0.562	0.000***	0.337	0.020**	0.002***	0.387	0.657
氧化钡	0.002***	0.700	0.779	0.395	0.001***	0.017**	0.040**	0.000***	0.337	0.000***	0.889	0.262	0.601	0.000***
五氧化二磷	0.000***	0.005***	0.470	0.000***	0.058	0.612	0.295	0.510	0.020**	0.889	0.000***	0.052*	0.576	0.235
氧化锶	0.000***	0.526	0.407	0.240	0.554	0.287	0.535	0.177	0.002***	0.262	0.052*	0.000***	0.796	0.213
氧化锡	0.560	0.573	0.180	0.148	0.089*	0.001***	0.119	0.207	0.387	0.601	0.576	0.796	0.000***	0.628
二氧化硫	0.006***	0.374	0.944	0.478	0.064*	0.155	0.216	0.096*	0.657	0.000***	0.235	0.213	0.628	0.000***

***: p<0.01 **: p<0.05 *: p<0.1

(b) 铅钡玻璃检验结果

图 15　Pearson 相关系数检验结果

图中标黄部分除对角线以外均为 p 值小于 0.05 的化学成分组合检验结果，其中高钾玻璃有 24 组化学成分具有显著相关关系，铅钡玻璃有 30 组化学成分具有显著相关关系，剩余组合的化学成分之间不存在显著相关关系。

5.10 问题四：关联关系差异性分析

对于任意两个不相同的化学成分，将前文得到的 Pearson 相关系数 r 和 Pearson 相关系数检验 p 值 p_r 进行组合，设定为新的向量 v。对于不同的玻璃类型，得到的向量可能存在差异，而向量 v 包含了固定玻璃类型下这两个化学成分的相关关系，那么只需要对比不同玻璃类型的对应向量的差异性，即可完成该问题。

对于不同玻璃类型的对应向量差异性的分析，由于得到的 Pearson 相关系数不服从正态分布，无法使用相关系数 t 检验，于是本文采用相关系数 z 检验进行分析，其原理和计算方法简述如下。

5.10.1 相关系数 z 检验

对于研究相关系数不满足正态分布时的不同相关系数之间的差异性，需要使用该相关系数 z 检验。该检验通过将相关系数进行 Fisher z 变换 [8]，将其转换成近似服从正态分布的变量，再利用该变量进行不同类型的统计分析。以下先介绍 Fisher z 变换的计算公式。

$$z = \frac{1}{2}\ln\left(\frac{1+r}{1-r}\right) \tag{16}$$

其中，r 为向量 v 的第一维，即 Pearson 相关系数。

针对得到的新变量 z，本文需要分析两个不同样本集之间的差异，那么对于同一组化学成分的相关系数变换后的 z 值进行分析。以下给出相关系数 z 检验的整体流程。

具体实现上，本文通过使用 Python 中的 norm.cdf 函数，求解 $|z_{value}/SE|$ 的 p 值，得到以下结果，见图 16。

根据图 16，图中标黄部分除对角线以外均为 p 值小于 0.05 的检验结果，共 27 组，根据其 p 值可以判断该组合的相关系数在不同玻璃类型当中体现出显著差异。接下来依据 图 15a和图 15b对图 16中的 27 组化学成分进行更加具体的关联关系的差异分析。

相关系数 z 检验流程
1、将不同玻璃类型得到的向量 v 的 r 进行 Fisher z 变换，得到 z_1, z_2
2、计算 z_1 和 z_2 的差 $z_{value} = z_1 - z_2$，和标准误 $SE = \sqrt{\frac{1}{n_1-3} + \frac{1}{n_2-3}}$，其中 n_1, n_2 为两个 Pearson 相关系数计算的样本量
3、查表得到给定显著性水平 α 下，标准正态分布的上 α 分位点 $z_{\alpha/2}$
4、判断 $\lvert z_{value}/SE \rvert > z_{\alpha/2}$ 是否成立，若成立，则两者存在显著差异

	二氧化硅	氧化钠	氧化钾	氧化钙	氧化镁	氧化铝	氧化铁	氧化铜	氧化铅	氧化钡	五氧化二磷	氧化锶	氧化锡	二氧化硫
二氧化硅	1.000***	0.003***	0.000***	0.034	0.008	0.000***	0.002	0.654	0.078*	0.717	0.593	0.919	0.901	0.911
氧化钠	0.003***	1.000***	0.022	0.000***	0.354	0.389	0.446	0.904	0.013**	0.59	0.438	0.760	0.936	0.823
氧化钾	0.000***	0.022	1.000***	0.003***	0.657	0.222	0.746	0.175	0.164	0.652	0.348	0.055*	0.855	0.259
氧化钙	0.034	0.000***	0.003***	1.000***	0.426	0.119	0.776	0.080*	0.964	0.494	0.035**	0.643	0.085*	0.233
氧化镁	0.008	0.354	0.657	0.426	1.000***	0.203	0.198	0.121	0.492	0.001***	0.095*	0.007***	0.956	0.018**
氧化铝	0.000***	0.389	0.222	0.119	0.203	1.000***	0.039**	0.091*	0.002***	0.011**	0.003***	0.005***	0.024**	0.230
氧化铁	0.002	0.446	0.746	0.776	0.198	0.039**	1.000***	0.005***	0.502	0.008***	0.004***	0.010***	0.128	0.159
氧化铜	0.654	0.904	0.175	0.080*	0.121	0.091*	0.005***	1.000***	0.398	0.271	0.686	0.975	0.519	0.826
氧化铅	0.078*	0.013**	0.164	0.964	0.492	0.002***	0.502	0.398	1.000***	0.003***	0.228	0.357	0.981	0.537
氧化钡	0.717	0.59	0.652	0.494	0.001***	0.011**	0.008***	0.271	0.003***	1.000***	0.160	0.110	0.887	0.001***
五氧化二磷	0.593	0.438	0.348	0.035**	0.095*	0.003***	0.004***	0.686	0.228	0.160	1.000***	0.025**	0.621	0.511
氧化锶	0.919	0.760	0.055*	0.643	0.007***	0.005***	0.010***	0.975	0.357	0.110	0.025**	1.000***	0.341	0.436
氧化锡	0.901	0.936	0.855	0.085*	0.956	0.024**	0.128	0.519	0.981	0.887	0.621	0.341	1.000***	0.901
二氧化硫	0.911	0.823	0.259	0.233	0.018**	0.230	0.159	0.826	0.537	0.001***	0.511	0.436	0.901	1.000***

***: p<0.01 **: p<0.05 *: p<0.1

图 16　相关系数 z 检验结果

5.10.2 具体差异分析

由于前文已经得到每一组化学成分的关联关系信息集合向量 v，本文从中挑出在图 16中 p 值小于 0.05 的集合向量。显然，这些集合向量当中应当包含同一组化学成分在不同玻璃类型当中的关联关系信息 v_1, v_2，其分别来自高钾玻璃的 Pearson 相关系数检验和铅钡玻璃的 Pearson 相关系数检验。本文从 v_1, v_2 中挑选出各自的相关系数检验的 p 值 p_{r1}, p_{r2}，通过对比两者之间的关系来分析差异。

首先根据 p_{r1}, p_{r2} 可以确定在不同玻璃类别当中，该组合的关联关系是否一致，即有 4 种可能的组合方式：

$$\begin{aligned} L_1 &: p_{r1} < 0.05 \& p_{r2} > 0.05 \\ L_2 &: p_{r1} > 0.05 \& p_{r2} < 0.05 \\ L_3 &: p_{r1} < 0.05 \& p_{r2} < 0.05 \\ L_4 &: p_{r1} > 0.05 \& p_{r2} > 0.05 \end{aligned} \tag{17}$$

其中 p_{r1} 为高钾玻璃的检验结果，p_{r2} 为铅钡玻璃的检验结果。

对于 L_1, L_2，可直接得出两个化学成分只在一种玻璃中存在显著相关关系，而在另一种玻璃中相关关系不显著的结论，相关关系的差异性明显；对于 L_3，可进一步对其相关系数 r_1, r_2 进行分析，若两者符号相反，这可得出两个化学成分在不同玻璃之间的关联关系相反的结论，若两者符号相同，则单独分析；对于 L_4，本文认为在两种玻璃之间都不存在显著相关关系的组合，但在不同玻璃之间却呈现出显著差异这是偶然现象，它可能具有一定的统计意义，但在实际生活当中没有实际意义，本文对这些组合不予考

虑。

经过比较、统计，得到以下结果，见表 22。

表 22　比较统计结果

L_1	K_2O/CaO Al_2O_3/P_2O_5 SiO_2/MgO	MgO/SrO Al_2O_3/SrO SiO_2/Fe_2O_3	Fe_2O_3/P_2O_5 Fe_2O_3/CuO	P_2O_5/SrO Fe_2O_3/SrO	Al_2O_3/Fe_2O_3 PbO/BaO	Na_2O/K_2O SiO_2/K_2O
L_2	SiO_2/Na_2O CaO/P_2O_5	Na_2O/PbO Al_2O_3/SnO_2	MgO/BaO BaO/SO_2	Al_2O_3/PbO	Al_2O_3/BaO	Fe_2O_3/BaO
L_3	Na_2O/CaO	SiO_2/Al_2O_3	SiO_2/CaO			
L_4	MgO/SO_2					

由表 22知：氧化钾与氧化钙的组合属于 L_1 类型，本文认为这可能与高钾玻璃的制作工艺有关，在制作时可能需要等比例的投放氧化钾和氧化钙，但是对于铅钡玻璃而言没有这一工序，所以导致了氧化钾和氧化钙在高钾玻璃里面的关联关系显著，在铅钡玻璃里面不显著的现象 [9]；二氧化硅与氧化铝的组合属于 L_3 类型中的相关系数符号相反的情况，本文通过查阅文献得知，氧化铝与二氧化硅在烧制玻璃时存在 $Al_2O_3+2SiO_2 \rightleftharpoons Al_2O_3 \cdot 2SiO_2$ 的反应，两种物质的增减趋势相同 [10]，也存在 $2Al_2O_3+Si_3N_4 \rightleftharpoons 3SiO_2+4AlN$ 的可逆反应，两种物质的增减趋势相反 [10]，对于不同的玻璃类型，其内在反应机理可能存在差异，会导致二氧化硅与氧化铝的关联关系出现相反的情况。

六、 模型的评价与改进与推广

6.1 模型的评价

6.1.1 模型的优点

1、本文在问题一对于统计规律的分析时，考虑不同数据的分布特征不同，针对性地使用不同的统计方法进行分析，考虑得比较全面。

2、问题 1 中对预测的多元线性回归和问题 2 中对亚类划分的层次聚类都进行了变量的筛选，可以去除掉干扰因素，使模型更加合理。

3、在进行聚类敏感性分析时，通过选定锚定点来定义指标的抗干扰能力，具有合理性和创新性。

4、本文使用 Logloss 损失函数对逻辑回归分类结果进行分析，可以表征分类结果在概率层面的准确率，比单纯使用预测准确率更能体现模型的优劣。

6.1.2 模型的不足

本文在对两种玻璃类型的亚类进行划分时，没有对模型的参数进行设置，但在敏感性分析当中得到参数对模型具有影响的结果，因此该模型的泛化性可能有待提高。

6.2 模型的改进

根据不足，在使用本文的分类模型进行实际分类任务时，可以进行训练参数的调整，根据 Logloss 损失函数值确定最合适的参数，使模型达到更好的效果。

6.3 模型的推广

1、本文针对古代玻璃进行研究的方法体系可以用于其他古代物品，如：瓷器、木制品、书画作品的研究分析当中。

2、本文对古代玻璃风化前的化学成分预测，可以应用与文物保护领域。对文物风化前后发生明显变化的化学成分进行针对性分析，以确定该文物的最佳保存环境，对文物保护工作具有一定的指导意义。

参考文献

[1] 王承遇 and 陶瑛, “硅酸盐玻璃的风化,” 硅酸盐学报, vol. 31, no. 1, pp. 78–85, 2003.

[2] 贾俊平 and 谭英平, 应用统计学. Gao deng jiao yu chu ban she, 2014.

[3] 邢航, “多因素方差分析中数学模型的建立与检验方法,” 电大理工, no. 2, pp. 73–75, 2008.

[4] 司守奎，孙玺菁, Python 数学实验与建模. SCIENCE PRESS, 2020.

[5] 王婕, 李沫, 马清林, 张治国, 章梅芳, 王菊琳 et al., “一件战国时期八棱柱状铅钡玻璃器的风化研究,” 玻璃与搪瓷, vol. 42, no. 2, pp. 6–13, 2014.

[6] 宋文平, 何明友, and 刘彪, “岩石物理化学在矿物共生组合研究中的应用,” 矿物学报, no. S1, pp. 672–673, 2015.

[7] Mechtool, “玻璃的分类和性能,” [EB/OL], https://www.mechtool.cn/nonmetalmaterial/nonmetalmaterial_roleofvariousoxidesinglass.html Accessed September 18, 2022.

[8] R. A. Fisher, “Frequency distribution of the values of the correlation coefficient in samples from an indefinitely large population,” Biometrika, vol. 10, no. 4, pp. 507–521, 1915.

[9] 李沫, “战国秦汉时期费昂斯制品的制备及铅钡玻璃研究,” Master’s thesis, 北京化工大学, 2014.

[10] S. B. Hong and M. A. Camblor, “Sio-defects in as-synthesized pure-silica and aluminosilicate sodalites,” Chemistry of materials, vol. 9, no. 9, pp. 1999–2003, 1997.

附录 A　附件目录结构

```
submit
├─1
│  ├─第一小问
│  │  ├─列联表分析
│  │  │  ├─纹饰和风化卡方.py
│  │  │  ├─纹饰和风化统计.py
│  │  │  ├─表面风化与类型卡方.py
│  │  │  ├─表面风化与类型统计.py
│  │  │  ├─附件.xlsx
│  │  │  ├─颜色和表面风化气泡图.py
│  │  │  └─颜色排序.xlsx
│  │  └─数据绘图
│  │  ├─有无风化
│  │  └─有无风化与类型
│  └─第二小问
│     ├─10 氧化钡.py
│     ├─11 五氧化二磷.py
│     ├─12 氧化锶.py
│     ├─13 氧化锡.py
│     ├─14 二氧化硫.py
│     ├─1 二氧化硅 _ 风化与类型.py
│     ├─2 氧化钠.py
│     ├─3 氧化钾.py
│     ├─4 氧化钙.py
│     ├─5 氧化镁.py
│     ├─6 氧化铝.py
│     ├─7 氧化铁.py
│     ├─8 氧化铜.py
│     ├─9 氧化铅.py
│     ├─第二小问结果整理.txt
│     └─表 1 表 2 整合.xlsx
```

```
├─2
│  ├─SPSS 聚类分析
│  │  ├─初始铅钡玻璃聚类分析.spv
│  │  ├─初始高钾玻璃聚类分析.spv
│  │  ├─初时肘部图.xlsx
│  │  ├─最终肘部图.xlsx
│  │  ├─最终铅钡玻璃聚类分析.spv
│  │  └─最终高钾玻璃聚类分析.spv
│  ├─敏感性分析
│  │  ├─铅钡玻璃二氧化硅.py
│  │  ├─铅钡玻璃五氧化二磷.py
│  │  ├─铅钡玻璃单个.py
│  │  ├─铅钡玻璃敏感性分析.txt
│  │  ├─铅钡玻璃氧化铅.py
│  │  ├─铅钡玻璃氧化锶.py
│  │  ├─铅钡筛选后数据.xlsx
│  │  ├─高钾抗干扰.xlsx
│  │  ├─高钾氧化铝.py
│  │  ├─高钾玻璃二氧化硅.py
│  │  ├─高钾玻璃敏感性分析.txt
│  │  ├─高钾玻璃氧化钙.py
│  │  ├─高钾玻璃氧化钾.py
│  │  ├─高钾玻璃氧化铜.py
│  │  ├─高钾玻璃氧化锡.py
│  │  └─高钾筛选后数据.xlsx
│  ├─方差分析
│  │  ├─铅钡
│  │  │  ├─方差分析.py
│  │  │  ├─铅钡方差分析结果.txt
│  │  │  └─铅钡聚类情况.xlsx
│  │  └─高钾
│  │     ├─方差分析.py
│  │     ├─高钾方差分析结果.txt
│  │     └─高钾聚类情况.xlsx
```

```
│   ├─轮廓系数计算
│   │   ├─轮廓系数铅钡.py
│   │   ├─轮廓系数高钾.py
│   │   ├─铅钡筛选后数据.xlsx
│   │   └─高钾筛选后数据.xlsx
│   └─逻辑回归
│       ├─logit.py
│       ├─分类表格整理.xlsx
│       ├─分类表格结果.xlsx
│       └─问题 3 数据.xlsx
├─3
│   ├─敏感性分析
│   │   ├─logit 超参数调整.py
│   │   ├─logit 输入调整
│   │   │   ├─10 氧化钡.py
│   │   │   ├─11 五氧化二磷.py
│   │   │   ├─12 氧化锶.py
│   │   │   ├─13 氧化锡.py
│   │   │   ├─14 二氧化硫.py
│   │   │   ├─1 二氧化硅.py
│   │   │   ├─2 氧化钠.py
│   │   │   ├─3 氧化钾.py
│   │   │   ├─4 氧化钙.py
│   │   │   ├─5 氧化镁.py
│   │   │   ├─6 氧化铝.py
│   │   │   ├─7 氧化铁.py
│   │   │   ├─8 氧化铜.py
│   │   │   ├─9 氧化铅.py
│   │   │   ├─分类表格整理.xlsx
│   │   │   ├─分类表格结果.xlsx
│   │   │   └─问题 3 数据.xlsx
│   │   ├─分类表格整理.xlsx
│   │   ├─分类表格结果.xlsx
│   │   └─问题 3 数据.xlsx
```

```
│   └─预测
│       ├─logit.py
│       ├─分类表格整理.xlsx
│       ├─分类表格结果.xlsx
│       └─问题 3 数据.xlsx
└─4
    ├─z 检验
    │   ├─z 检验.py
    │   ├─z 检验 p 值结果.csv
    │   ├─铅钡总表.xlsx
    │   └─高钾总表.xlsx
    └─皮尔逊相关系数检验
        ├─temp.png
        ├─铅钡总表.xlsx
        ├─铅钡指标间热图.py
        ├─铅钡检验结果.txt
        ├─铅钡热图 p 值.csv
        ├─铅钡热图相关系数.csv
        ├─高钾总表.xlsx
        ├─高钾检验结果.txt
        ├─高钾热图.png
        ├─高钾热图.py
        └─高钾热图 p 值.csv
```

附录 B　问题一第三小问预测分析结果

文物采样点	二氧化硅 (SiO2)	氧化钠 (Na2O)	氧化钾 (K2O)	氧化钙 (CaO)	氧化镁 (MgO)
07	76.11237	1.330863	6.444558	5.409024	0.932646
09	76.34831	1.652598	7.795843	4.818522	1.017709
10	76.62203	1.992194	8.699623	4.635641	1.033292
12	76.70063	1.433413	7.314745	4.303029	1.123115
22	77.91938	0.98157	6.789297	4.510746	1.330054
27	77.83158	1.386528	5.612907	5.410869	1.200738
02	39.9814	1.820958	0.589563	1.903181	1.372815
08	44.64114	0.518485	0.116172	0.013137	0
08 严重风化点	38.2225	1.006227	0.106033	0.009238	0
11	37.66659	0.919218	0.143173	1.802946	0.329264
19	36.8461	1.601549	0.314902	1.541082	0.748154
26	43.97362	0.46429	0.086091	0	0
26 严重风化点	39.08751	0	0.109199	0.18507	0
34	42.59977	2.76382	0.149177	0	0.165428
36	45.15883	2.886348	0.143048	0	0.11353
38	38.20819	2.269038	0.190261	0	0.418791
39	31.31621	1.527443	0.075494	0	0
40	30.12877	0.576547	0.069857	0.015516	0

41	37.63576	1.078941	0.495464	2.741429	0.626479
43 部位 1	29.69803	0	0.172013	3.44071	0.545863
43 部位 2	39.56786	0.911036	0.2996	1.358879	0.788163
48	60.0705	1.468676	0.458687	1.260303	1.050204
49	36.629	1.286089	0.633153	1.627351	1.10464
50	36.0122	1.60875	0.133992	1.238808	0.140678
51 部位 1	37.65731	1.783887	0.448657	1.986861	0.950847
51 部位 2	36.80429	1.034987	0.211073	2.130619	0.638581
52	37.60608	1.47612	0.087093	1.346044	0.013172
54	29.8469	1.593934	0.203986	2.695771	0.85396
54 严重风化点	51.25683	0.645187	0.182766	1.087648	0.938039
56	43.81324	1.486529	0.135967	0	0.109247
57	46.47588	2.076594	0.146802	0	0.168838
58	36.77137	1.371516	0.272907	1.713722	0.665539

文物采样点	氧化铝 (Al2O3)	氧化铁 (Fe2O3)	氧化铜 (CuO)	氧化铅 (PbO)	氧化钡 (BaO)
07	3.355341	1.475225	0.348346	0	0.003017
09	3.146438	2.0111	0	0	0.014384
10	2.590365	2.76241	0	0	0.007353
12	2.894809	2.28146	0	0	0.014814
22	3.113815	2.09666	0	0	0.002471
27	3.025836	1.953538	0.231687	0	0.014334
02	9.681552	1.552264	0.766879	19.7	6.004837
08	5.736608	0.614232	8.53141	23.9	31.85542
08 严重风化点	6.10902	1.495644	4.973236	16.8	35.38804
11	6.531828	1.177859	2.397161	26.5	12.00451
19	8.181683	1.112038	0.862729	21.9	11.99983
26	5.820961	0.595633	9.334465	25.9	31.30993
26 严重风化点	5.859588	1.583867	8.456601	18.8	43.59711
34	7.462549	0.679065	0.257901	27.5	12.21627
36	6.995395	0.624413	0.55163	27.7	9.501317
38	7.720235	0.67956	0.14699	26.9	9.67006
39	8.787986	0.760426	0.099018	29.7	10.1723
40	9.594395	0.765412	0.397676	29.3	7.140335
41	11.61098	2.275829	0	20.8	9.510221
43 部位 1	10.46089	1.731161	1.626692	24	9.879588
43 部位 2	9.004565	1.931136	1.63916	22.7	7.476203
48	20.70498	1.254975	0.20976	4.73	6.183959
49	8.648925	1.956367	0.596144	22.1	6.103221
50	8.088672	1.049709	1.188871	28	8.748821
51 部位 1	10.18058	1.399763	1.147393	18.1	6.628274
51 部位 2	10.56341	2.259456	1.976339	17.5	6.10815
52	8.57683	0.928315	0.853567	27.2	7.554737
54	10.71949	1.371084	1.106604	34.5	6.253763
54 严重风化点	10.73007	1.241225	2.927462	43.6	6.829938
56	6.961053	0.613116	0.982161	25.2	9.871673
57	7.326219	0.609062	1.229825	21.9	11.09377
58	8.161955	1.264314	0.90788	22.9	9.872538

文物采样点	五氧化二磷 (P2O5)	氧化锶 (SrO)	氧化锡 (SnO2)	二氧化硫 (SO2)
07	1.004207	0.015109	1.06163	0
09	0.997028	0.025119	0.966145	0
10	1.07774	0.041817	1.176902	0
12	1.000171	0.034748	0.916202	0
22	0.770481	0.040861	1.075206	0
27	0.928386	0.034996	1.196128	0
02	2.567903	0.488097	0.217224	0
08	1.833887	0.545977	0	3.814441
08 严重风化点	0	0.19848	0	8.064019
11	3.946381	0.350062	0	0.819833
19	3.247865	0.485476	0.036144	0
26	1.809282	0.538067	0	3.946429
26 严重风化点	0	0.293348	0	9.329452
34	0.918699	0.481608	0	0.297297
36	0	0.425252	0	0.196625
38	0.145565	0.515707	0	0.361357

39	1.249578	0.680481	0	0.035485
40	2.04832	0.832312	0	0
41	5.823137	0.479627	0	0.941289
43 部位 1	6.207015	0.669133	0	0.384817
43 部位 2	7.871407	0.590131	0.006339	0
48	2.229908	0.217985	0.629886	0
49	5.313275	0.449229	0.220619	0.058212
50	3.557305	0.46567	0	2.103114
51 部位 1	4.032701	0.447852	0.11696	0.693634
51 部位 2	6.055377	0.582519	0	0
52	1.350165	0.499341	0	0.509197
54	3.557305	0.595468	0.030583	0.301221
54 严重风化点	0.174996	0.791671	0.00467	0
56	1.35175	0.41549	0	1.139478
57	1.454809	0.465934	0	1.723666
58	3.921798	0.453228	0.002759	0.305499

附录 C　问题二第二小问方差分析结果

二氧化硅	df	sum_sq	mean_sq	F	PR(>F)
种类	1	3773.311	3773.311	55.07103	1.89E-09
Residual	47	3220.307	68.51718	NaN	NaN

氧化钠	df	sum_sq	mean_sq	F	PR(>F)
种类	1	8.860923	8.860923	3.208402	0.079702
Residual	47	129.804	2.761787	NaN	NaN

氧化钾	df	sum_sq	mean_sq	F	PR(>F)
种类	1	0.035833	0.035833	0.61344	0.437424
Residual	47	2.745416	0.058413	NaN	NaN

氧化钙	df	sum_sq	mean_sq	F	PR(>F)
种类	1	1.098871	1.098871	0.813509	0.371684
Residual	47	63.48656	1.350778	NaN	NaN

氧化镁	df	sum_sq	mean_sq	F	PR(>F)
种类	1	0.007548	0.007548	0.030771	0.861507
Residual	47	11.52876	0.245293	NaN	NaN

氧化铝	df	sum_sq	mean_sq	F	PR(>F)
种类	1	2.177367	2.177367	0.149025	0.701211
Residual	47	686.7067	14.61078	NaN	NaN

氧化铁	df	sum_sq	mean_sq	F	PR(>F)
种类	1	0.44403	0.44403	0.541783	0.465353
Residual	47	38.51991	0.819573	NaN	NaN

氧化铜	df	sum_sq	mean_sq	F	PR(>F)
种类	1	2.531054	2.531054	0.442724	0.509063
Residual	47	268.6993	5.717006	NaN	NaN

氧化铅	df	sum_sq	mean_sq	F	PR(>F)
种类	1	1026.05	1026.05	28.22985	0.000003
Residual	47	1708.275	36.34628	NaN	NaN

氧化钡	df	sum_sq	mean_sq	F	PR(>F)
种类	1	5.714185	5.714185	0.075631	0.784513
Residual	47	3550.991	75.55301	NaN	NaN

五氧化二磷	df	sum_sq	mean_sq	F	PR(>F)
种类	1	25.90461	25.90461	5.966627	0.018393
Residual	47	204.0544	4.341584	NaN	NaN

氧化锶	df	sum_sq	mean_sq	F	PR(>F)
种类	1	0.603163	0.603163	14.99252	0.000332
Residual	47	1.890854	0.040231	NaN	NaN

氧化锡	df	sum_sq	mean_sq	F	PR(>F)
种类	1	0.004368	0.004368	0.256808	0.614691
Residual	47	0.799451	0.01701	NaN	NaN

二氧化硫	df	sum_sq	mean_sq	F	PR(>F)
种类	1	3.161989	3.161989	0.856715	0.359388
Residual	47	173.4689	3.690829	NaN	NaN

附录 D　问题二第二小问聚类结果

文物采样点	二氧化硅 (SiO2)	氧化钠 (Na2O)	氧化钾 (K2O)	氧化钙 (CaO)	氧化镁 (MgO)
07	76.11237	1.330863	6.444558	5.409024	0.932646
09	76.34831	1.652598	7.795843	4.818522	1.017709
10	76.62203	1.992194	8.699623	4.635641	1.033292
12	76.70063	1.433413	7.314745	4.303029	1.123115
22	77.91938	0.98157	6.789297	4.510746	1.330054
27	77.83158	1.386528	5.612907	5.410869	1.200738
03 部位 1	87.05	0	5.19	2.01	0
18	79.46	0	9.42	0	1.53
21	76.68	0	0	4.71	1.22
01	69.33	0	9.99	6.32	0.87
03 部位 2	61.71	0	12.37	5.87	1.11
04	65.88	0	9.67	7.12	1.56
05	61.58	0	10.95	7.35	1.77
06 部位 1	67.65	0	7.37	0	1.98
06 部位 2	59.81	0	7.68	5.41	1.73
13	59.01	2.86	12.53	8.7	0
14	62.47	3.38	12.28	8.23	0.66
16	65.18	2.1	14.52	8.27	0.52
文物采样点	氧化铝 (Al2O3)	氧化铁 (Fe2O3)	氧化铜 (CuO)	氧化铅 (PbO)	氧化钡 (BaO)
07	3.355341	1.475225	0.348346	0	0.003017
09	3.146438	2.0111	0	0	0.014384
10	2.590365	2.76241	0	0	0.007353
12	2.894809	2.28146	0	0	0.014814
22	3.113815	2.09666	0	0	0.002471
27	3.025836	1.953538	0.231687	0	0.014334
03 部位 1	4.06	0	0.78	0.25	0
18	3.05	0	0	0	0
21	6.19	2.37	3.28	1	1.97
01	3.93	1.74	3.87	0	0
03 部位 2	5.5	2.16	5.09	1.41	2.86
04	6.44	2.06	2.18	0	0
05	7.5	2.62	3.27	0	0
06 部位 1	11.15	2.39	2.51	0.2	1.38
06 部位 2	10.05	6.04	2.18	0.35	0.97
13	6.16	2.88	4.73	0	0
14	9.23	0.5	0.47	1.62	0
16	6.18	0.42	1.07	0.11	0
文物采样点	五氧化二磷 (P2O5)	氧化锶 (SrO)	氧化锡 (SnO2)	二氧化硫 (SO2)	类型
07	1.004207	0.015109	1.06163	0	高钾
09	0.997028	0.025119	0.966145	0	高钾
10	1.07774	0.041817	1.176902	0	高钾
12	1.000171	0.034748	0.916202	0	高钾
22	0.770481	0.040861	1.075206	0	高钾
27	0.928386	0.034996	1.196128	0	高钾
03 部位 1	0.66	0	0	0	高钾
18	1.36	0.07	2.36	0	高钾

21	1.1	0	0	0	高钾
01	1.17	0	0	0.39	高钾
03 部位 2	0.7	0.1	0	0	高钾
04	0.79	0	0	0.36	高钾
05	0.94	0.06	0	0.47	高钾
06 部位 1	4.18	0.11	0	0	高钾
06 部位 2	4.5	0.12	0	0	高钾
13	1.27	0	0	0	高钾
14	0.16	0	0	0	高钾
16	0	0.04	0	0	高钾

文物采样点	亚类划分
07	1
09	1
10	1
12	1
22	1
27	1
03 部位 1	1
18	1
21	1
01	2
03 部位 2	2
04	2
05	2
06 部位 1	2
06 部位 2	2
13	2
14	2
16	2

文物采样点	二氧化硅 (SiO2)	氧化铅 (PbO)	五氧化二磷 (P2O5)	氧化锶 (SrO)	聚类结果
2	39.98	19.7	2.57	0.49	1
11	37.67	26.5	3.95	0.35	1
19	36.85	21.9	3.25	0.49	1
34	42.6	27.5	0.92	0.48	1
36	45.16	27.7	0	0.43	1
38	38.21	26.9	0.15	0.52	1
39	31.32	29.7	1.25	0.68	1
40	30.13	29.3	2.05	0.83	1
41	37.64	20.8	5.82	0.48	1
43 部位 1	29.7	24	6.21	0.67	1
43 部位 2	39.57	22.7	7.87	0.59	1
49	36.63	22.1	5.31	0.45	1
50	36.01	28	3.56	0.47	1
51 部位 1	37.66	18.1	4.03	0.45	1
51 部位 2	36.8	17.5	6.06	0.58	1
52	37.61	27.2	1.35	0.5	1
54	29.85	34.5	3.56	0.6	1
56	43.81	25.2	1.35	0.42	1
57	46.48	21.9	1.45	0.47	1
58	36.77	22.9	3.92	0.45	1
23 未风化点	53.79	16.98	0	0.33	1
25 未风化点	50.61	31.9	0.19	0.2	1
42 未风化点 1	51.26	21.88	0.08	0.35	1
42 未风化点 2	51.33	20.12	0	0	1
46	55.21	25.25	0.2	0.43	1
47	51.54	25.4	0.1	0.85	1
49 未风化点	54.61	23.02	4.32	0.3	1
50 未风化点	45.02	30.61	6.34	0.23	1
55	49.01	32.92	0.35	0	1
8	44.64	23.9	1.83	0.55	2
08 严重风化点	38.22	16.8	0	0.2	2
26	43.97	25.9	1.81	0.54	2
26 严重风化点	39.09	18.8	0	0.29	2
20	37.36	9.3	5.75	0	2

24	31.94	29.14	0.14	0.91	2
54 严重风化点	51.26	43.6	0.17	0.79	1
30 部位 1	34.34	39.22	0	0.35	1
30 部位 2	36.93	37.74	1.41	0.48	1
48	60.07	4.73	2.23	0.22	3
28 未风化点	68.08	17.14	1.04	0.12	3
29 未风化点	63.3	12.31	0.41	0.25	3
31	65.91	16.55	1.62	0.3	3
32	69.71	19.76	0.17	0	3
33	75.51	16.16	0.13	0	3
35	65.91	22.05	0.42	0	3
37	60.12	17.24	1.46	0.31	3
44 未风化点	60.74	13.61	0	0.26	3
45	61.28	15.99	0	0.23	3
53 未风化点	63.66	13.66	0	0.27	3

文物采样点	亚类划分
2	1
11	1
19	1
34	1
36	1
38	1
39	1
40	1
41	1
43 部位 1	1
43 部位 2	1
49	1
50	1
51 部位 1	1
51 部位 2	1
52	1
54	1
56	1
57	1
58	1
23 未风化点	1
25 未风化点	1
42 未风化点 1	1
42 未风化点 2	1
46	1
47	1
49 未风化点	1
50 未风化点	1
55	1
8	1
08 严重风化点	1
26	1
26 严重风化点	1
20	1
24	1
54 严重风化点	2
30 部位 1	2
30 部位 2	2
48	3
28 未风化点	3
29 未风化点	3
31	3
32	3
33	3
35	3
37	3
44 未风化点	3
45	3

53 未风化点　　3

附录 E　主要代码

纹饰和风化类型统计

```
import pandas as pd

data = pd.read_excel("附件.xlsx", sheet_name="表单1")
data = data.values
a = 0 # 无风化 a
b = 0 # 无风化 b
c = 0 # 无风化 c
d = 0 # 风化 a
e = 0 # 风化 b
f = 0 # 风化 c

for i in range(data.shape[0]):
    if (data[i, 4] == "无风化" and data[i, 1] == "A"):
        a += 1
    elif (data[i, 4] == "无风化" and data[i, 1] == "B"):
        b += 1
    elif (data[i, 4] == "无风化" and data[i, 1] == "C"):
        c += 1
    elif(data[i, 4] == "风化" and data[i, 1] == "A"):
        d += 1
    elif(data[i, 4] == "风化" and data[i, 1] == "B"):
        e+=1
    else:
        f+=1
```

纹饰和风化卡方检验

```
import math

import numpy as np
import pandas as pd
import scipy.stats as stats

table = np.zeros([3, 2])
table[0, 0] = 11
table[0, 1] = 11
table[1, 0] = 0
table[1, 1] = 6
table[2, 0] = 13
table[2, 1] = 17

kafang, p, ziyoudu, pingshubiao = stats.chi2_contingency(table)

xinagguanxishu = math.sqrt(kafang / (kafang + 58))
print("卡方值", kafang)
print("p值", p)
print("自由度", ziyoudu)
print("频数表", pingshubiao)
print("相关系数", xinagguanxishu)
```

RankBiserial 秩相关分析与气泡图绘制

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from scipy import stats
```

```
data = pd.read_excel("颜色排序.xlsx")
data = data.values

x = data[:, 4]
y = data[:, 3]

# print(stats.pointbiserialr(x, y))
# y是颜色
# x是二值


mydata = {'x': y, 'y': x}

df=pd.DataFrame(mydata)
my_mean=df.groupby('y')['x'].mean()

r_pb = 2*(my_mean[1]-my_mean[0])/(df.shape[0])

print(r_pb)

plotx = np.array([0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1])
ploty = np.array([1, 2, 3, 4, 5, 6, 7, 1, 3, 4, 5, 7, 8])
z = np.zeros(13)

for i in range(54):
    if (x[i] == 0 and y[i] == 1):
        z[0] += 1
    elif (x[i] == 0 and y[i] == 2):
        z[1] += 1
    elif (x[i] == 0 and y[i] == 3):
        z[2] += 1
    elif (x[i] == 0 and y[i] == 4):
        z[3] += 1
    elif (x[i] == 0 and y[i] == 5):
        z[4] += 1
    elif (x[i] == 0 and y[i] == 6):
        z[5] += 1
    elif (x[i] == 0 and y[i] == 7):
        z[6] += 1
    elif (x[i] == 1 and y[i] == 1):
        z[7] += 1
    elif (x[i] == 1 and y[i] == 3):
        z[8] += 1
    elif (x[i] == 1 and y[i] == 4):
        z[9] += 1
    elif (x[i] == 1 and y[i] == 5):
        z[10] += 1
    elif (x[i] == 1 and y[i] == 7):
        z[11] += 1
    elif (x[i] == 1 and y[i] == 8):
        z[12] += 1
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

plt.xlabel('表面是否风化（0为无风化，1为风化）', fontsize=13)
plt.ylabel('颜色（从1至8依次加深）', fontsize=13)

plt.scatter(plotx, ploty, s=250 * z, alpha=0.5)
plt.xlim(-1, 2)
plt.ylim(0, 9)
plt.show()
```

第一问第二小问二氧化硅多因素方差分析

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import scipy
import scipy.stats as stats
from statsmodels.stats.anova import anova_lm
from statsmodels.formula.api import ols
from statsmodels.stats.multicomp import pairwise_tukeyhsd
import statsmodels as sm

data = pd.read_excel("表1表2整合.xlsx")

data = data.values

a = 12 # 无风化 高钾
b = 23 # 无风化 铅钡
c = 6 # 风化 高钾
d = 26 # 风化 铅钡

alist = [0, 2, 3, 4, 5, 6, 7, 15, 16, 17, 18, 21]
blist = [20, 23, 24, 25, 29, 30, 31, 32, 33, 34, 35, 37, 39, 44, 45, 48, 49, 50, 51, 54, 56,
    60, 63]
clist = [8, 11, 12, 14, 22, 28]
dlist = [1, 9, 10, 13, 19, 26, 27, 36, 38, 40, 41, 42, 43, 46, 47, 52, 53, 55, 57, 58, 59, 61,
    62, 64, 65, 66]
#
# for i in range(data.shape[0]):
#     if(data[i,16]=="无风化" and data[i,18]=="高钾"):
#         a+=1
#         alist.append(i)
#     elif(data[i,16]=="无风化" and data[i,18]=="铅钡"):
#         b+=1
#         blist.append(i)
#     elif(data[i,16]=="风化" and data[i,18]=="高钾"):
#         c+=1
#         clist.append(i)
#     else:
#         d+=1
#         dlist.append(i)

ya = np.zeros(a)
yb = np.zeros(b)
yc = np.zeros(c)
yd = np.zeros(d)

idx = 1

for i in range(a):
    ya[i] = data[alist[i], idx]

for i in range(b):
    yb[i] = data[blist[i], idx]

for i in range(c):
    yc[i] = data[clist[i], idx]

for i in range(d):
    yd[i] = data[dlist[i], idx]

ya = np.sort(ya)
yb = np.sort(yb)
yc = np.sort(yc)
yd = np.sort(yd)
```

```
def self_JBtest(y):
    # 样本规模n
    n = y.size
    # for i in range(data.shape[0]):
    #     if(data[i,16]=="无风化"):
    #         n+=1
    y_ = y - y.mean()
    M2 = np.mean(y_ ** 2)

    skew = np.mean(y_ ** 3) / M2 ** 1.5
    krut = np.mean(y_ ** 4) / M2 ** 2

    #         wufenglist.append(i)
    #     else:
    #         m+=1
    #         fenglist.append(i)
    JB = n * (skew ** 2 / 6 + (krut - 3) ** 2 / 24)
    pvalue = 1 - stats.chi2.cdf(JB, df=2)
    print("JB检验：", stats.jarque_bera(y))
    return np.array([JB, pvalue])


self_JBtest(ya)
self_JBtest(yb)
self_JBtest(yc)
self_JBtest(yd)

print('基于中位数的levene test P值：', stats.levene(ya, yb, yc, yd, center='median').pvalue)

predata = np.zeros([67, 3])

predata[:, 0] = data[:, idx]

for i in range(67):
    if (data[i, 18] == "高钾"):
        predata[i, 2] = 0
    else:
        predata[i, 2] = 1
    if (data[i, 16] == "无风化"):
        predata[i, 1] = 0
    else:
        predata[i, 1] = 1

predata = pd.DataFrame(predata, columns=['二氧化硅', '风化', '类型'])

model = ols("二氧化硅 ~风化+类型+风化:类型", data=predata)
mydata = model.fit()
print(anova_lm(mydata))
```

第一问第二小问氧化钠分析

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import scipy
import scipy.stats as stats
from statsmodels.stats.anova import anova_lm
from statsmodels.formula.api import ols
from statsmodels.stats.multicomp import pairwise_tukeyhsd
import statsmodels as sm

data = pd.read_excel("表1表2整合.xlsx")
```

```
data = data.values

a = 12 # 无风化 高钾
b = 23 # 无风化 铅钡
c = 6 # 风化 高钾
d = 26 # 风化 铅钡

alist = [0, 2, 3, 4, 5, 6, 7, 15, 16, 17, 18, 21]
blist = [20, 23, 24, 25, 29, 30, 31, 32, 33, 34, 35, 37, 39, 44, 45, 48, 49, 50, 51, 54, 56,
    60, 63]
clist = [8, 11, 12, 14, 22, 28]
dlist = [1, 9, 10, 13, 19, 26, 27, 36, 38, 40, 41, 42, 43, 46, 47, 52, 53, 55, 57, 58, 59, 61,
    62, 64, 65, 66]
#
# for i in range(data.shape[0]):
#     if(data[i,16]=="无风化" and data[i,18]=="高钾"):
#         a+=1
#         alist.append(i)
#     elif(data[i,16]=="无风化" and data[i,18]=="铅钡"):
#         b+=1
#         blist.append(i)
#     elif(data[i,16]=="风化" and data[i,18]=="高钾"):
#         c+=1
#         clist.append(i)
#     else:
#         d+=1
#         dlist.append(i)

ya = np.zeros(a)
yb = np.zeros(b)
yc = np.zeros(c)
yd = np.zeros(d)

idx = 2

for i in range(a):
    ya[i] = data[alist[i], idx]

for i in range(b):
    yb[i] = data[blist[i], idx]

for i in range(c):
    yc[i] = data[clist[i], idx]

for i in range(d):
    yd[i] = data[dlist[i], idx]

ya = np.sort(ya)
yb = np.sort(yb)
yc = np.sort(yc)
yd = np.sort(yd)


def self_JBtest(y):
    # 样本规模n
    n = y.size
    # for i in range(data.shape[0]):
    #     if(data[i,16]=="无风化"):
    #         n+=1
    y_ = y - y.mean()
    M2 = np.mean(y_ ** 2)

    skew = np.mean(y_ ** 3) / M2 ** 1.5
    krut = np.mean(y_ ** 4) / M2 ** 2
```

```
    #         wufenglist.append(i)
    #     else:
    #         m+=1
    #         fenglist.append(i)
    JB = n * (skew ** 2 / 6 + (krut - 3) ** 2 / 24)
    pvalue = 1 - stats.chi2.cdf(JB, df=2)
    print("JB检验：", stats.jarque_bera(y))
    return np.array([JB, pvalue])


self_JBtest(ya)
self_JBtest(yb)
self_JBtest(yc)
self_JBtest(yd)

print('基于中位数的levene test P值：', stats.levene(ya, yb, yc, yd, center='median').pvalue)

predata = np.zeros([67, 3])

predata[:, 0] = data[:, idx]

for i in range(67):
    if (data[i, 18] == "高钾"):
        predata[i, 2] = 0
    else:
        predata[i, 2] = 1
    if (data[i, 16] == "无风化"):
        predata[i, 1] = 0
    else:
        predata[i, 1] = 1

predata = pd.DataFrame(predata, columns=['氧化钠', '风化', '类型'])

model = ols("氧化钠 ~风化+类型+风化:类型", data=predata)
mydata = model.fit()
print(anova_lm(mydata))



gaojia=np.append(ya,yc)
qianbei=np.append(yb,yd)


fenghua=np.append(ya,yb)
wufenghua=np.append(yc,yd)


import scipy.stats as ss
from scipy.stats import ttest_ind
print("分为有无风化 组间比较 非参数检验")
print(ss.ranksums(wufenghua, fenghua))
print("无风化内部 做t检验")
print(ttest_ind(ya,yb))

self_JBtest(ya)
self_JBtest(yb)

self_JBtest(wufenghua)
self_JBtest(fenghua)

print(np.mean(ya),np.mean(yb),np.mean(ya)-np.mean(yb))
```

第一问第三小问铅钡热图绘制

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import spearmanr, pearsonr

plt.rcParams['font.sans-serif'] = ['FangSong']
plt.rcParams['axes.unicode_minus'] = False
z = pd.read_excel("铅钡总表.xlsx")
z = z.values

data = z[:, :]

res = pd.DataFrame(data, columns=[
    '二氧化硅', '氧化钠', '氧化钾', '氧化钙', '氧化镁', '氧化铝', '氧化铁', '氧化铜', '氧化铅',
    '氧化钡', '五氧化二磷', '氧化锶', '氧化锡', '二氧化硫'], dtype=np.float)

corr = res.corr(method='pearson')

sns.heatmap(corr, annot=True, vmax=1, vmin=-1, xticklabels=True, yticklabels=True,
            square=True,
                cmap="RdYlGn",annot_kws={"fontsize":7}).get_figure().savefig("temp.png",dpi=500,bbox_inches
                = 'tight')

plt.show()
```

第一问第三小问铅钡氧化钠拟合

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from scipy.optimize import curve_fit

plt.rcParams['font.sans-serif'] = ['FangSong']

data = pd.read_excel("铅钡氧化钠.xlsx")

data = data.values

y = data[:, 0]

y = y.flatten()

x = np.zeros([5, y.shape[0]])
for i in range(5):
    x[i] = data[:, i + 1]


def fun(x, px11, px22, px33, pt1, pt2, px1, px2, px3, px12, px13, px23, px00):
    x1 = x[0, :]
    x2 = x[1, :]
    x3 = x[2, :]
    t1 = x[3, :]
    t2 = x[4, :]

    return px11 * x1 ** 2 + px22 * x2 ** 2 \
           + px33 * x3 ** 2 + pt1 * t1 + pt2 * t2 \
           + px1 * x1 + px2 * x2 + px3 * x3 + px12 * x1 * x2 \
           + px13 * x1 * x3 + px23 * x2 * x3 + px00


def funlin(x, pt1, pt2, px1, px2, px3, px00):
    x1 = x[0, :]
```

```
    x2 = x[1, :]
    x3 = x[2, :]
    t1 = x[3, :]
    t2 = x[4, :]
    return px1 * x1 + px2 * x2 + px3 * x3 + px00 + pt1 * t1 + pt2 * t2


popt, pcov = curve_fit(fun, x, y,method='trf')

px11, px22, px33, pt1, pt2, px1, px2, px3, px12, px13, px23, px00 = popt

y2 = np.zeros(y.shape[0])

former=[]
latter=[]


for i in range(y2.shape[0]):
    x1 = x[0, i]
    x2 = x[1, i]
    x3 = x[2, i]
    t1 = x[3, i]
    t2 = x[4, i]
    y2[i] = px11 * x1 ** 2 + px22 * x2 ** 2 + px33 * x3 ** 2 + pt1 * t1 + pt2 * t2 + px1 * x1 +
        px2 * x2 + px3 * x3 + px12 * x1 * x2 + px13 * x1 * x3 + px23 * x2 * x3 + px00
    if(t1==1 or t2==1):
        t1=0
        t2=0
        former.append(y[i])
        latter.append(px11 * x1 ** 2 + px22 * x2 ** 2 + px33 * x3 ** 2 + pt1 * t1 + pt2 * t2 +
            px1 * x1 + px2 * x2 + px3 * x3 + px12 * x1 * x2 + px13 * x1 * x3 + px23 * x2 * x3 +
            px00)


def __sst(y_no_fitting):
    """
    计算SST(total sum of squares) 总平方和
    :param y_no_predicted: List[int] or array[int] 待拟合的y
    :return: 总平方和SST
    """
    y_mean = sum(y_no_fitting) / len(y_no_fitting)
    s_list = [(y - y_mean) ** 2 for y in y_no_fitting]
    sst = sum(s_list)
    return sst


def __ssr(y_fitting, y_no_fitting):
    """
    计算SSR(regression sum of squares) 回归平方和
    :param y_fitting: List[int] or array[int] 拟合好的y值
    :param y_no_fitting: List[int] or array[int] 待拟合y值
    :return: 回归平方和SSR
    """
    y_mean = sum(y_no_fitting) / len(y_no_fitting)
    s_list = [(y - y_mean) ** 2 for y in y_fitting]
    ssr = sum(s_list)
    return ssr


def __sse(y_fitting, y_no_fitting):
    """
    计算SSE(error sum of squares) 残差平方和
    :param y_fitting: List[int] or array[int] 拟合好的y值
    :param y_no_fitting: List[int] or array[int] 待拟合y值
```

```
    :return: 残差平方和SSE
    """
    s_list = [(y_fitting[i] - y_no_fitting[i]) ** 2 for i in range(len(y_fitting))]
    sse = sum(s_list)
    return sse


def goodness_of_fit(y_fitting, y_no_fitting):
    """
    计算拟合优度R^2
    :param y_fitting: List[int] or array[int] 拟合好的y值
    :param y_no_fitting: List[int] or array[int] 待拟合y值
    :return: 拟合优度R^2
    """
    SSR = __ssr(y_fitting, y_no_fitting)
    SST = __sst(y_no_fitting)
    rr = SSR / SST
    return rr


print(goodness_of_fit(y2, y))


print(former)
print(latter)

ans=np.zeros([len(former),2])

ans[:,0]=np.array(former)
ans[:,1]=np.array(latter)
np.savetxt("铅钡氧化钠预测.csv",ans,delimiter=',')

from scipy import stats

ci = 0.95
pp = (1. + ci) / 2.
nstd = stats.norm.ppf(pp)
perr = np.sqrt(np.diag(pcov))
popt_up = popt + nstd * perr
popt_dw = popt - nstd * perr
print("置信区间")
print(popt_dw)
print(popt_up)

print(popt)
```

第一问第三小问高钾氧化铜拟合

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from scipy.optimize import curve_fit

plt.rcParams['font.sans-serif'] = ['FangSong']

data = pd.read_excel("高钾氧化铜.xlsx")

data = data.values

y = data[:, 0]

y = y.flatten()

x = np.zeros([5, y.shape[0]])
```

```
for i in range(5):
    x[i] = data[:, i + 1]


def fun(x, px11, px22, px33, pt1, pt2, px1, px2, px3, px12, px13, px23, px00):
    x1 = x[0, :]
    x2 = x[1, :]
    x3 = x[2, :]
    t1 = x[3, :]
    t2 = x[4, :]

    return px11 * x1 ** 2 + px22 * x2 ** 2 \
          + px33 * x3 ** 2 + pt1 * t1 + pt2 * t2 \
          + px1 * x1 + px2 * x2 + px3 * x3 + px12 * x1 * x2 \
          + px13 * x1 * x3 + px23 * x2 * x3 + px00


def funlin(x, pt1, pt2, px1, px2, px3, px00):
    x1 = x[0, :]
    x2 = x[1, :]
    x3 = x[2, :]
    t1 = x[3, :]
    t2 = x[4, :]
    return px1 * x1 + px2 * x2 + px3 * x3 + px00 + pt1 * t1 + pt2 * t2


popt, pcov = curve_fit(fun, x, y,method='trf')

px11, px22, px33, pt1, pt2, px1, px2, px3, px12, px13, px23, px00 = popt

y2 = np.zeros(y.shape[0])

former=[]
latter=[]


for i in range(y2.shape[0]):
    x1 = x[0, i]
    x2 = x[1, i]
    x3 = x[2, i]
    t1 = x[3, i]
    t2 = x[4, i]
    y2[i] = px11 * x1 ** 2 + px22 * x2 ** 2 + px33 * x3 ** 2 + pt1 * t1 + pt2 * t2 + px1 * x1 +
        px2 * x2 + px3 * x3 + px12 * x1 * x2 + px13 * x1 * x3 + px23 * x2 * x3 + px00
    if(t1==1 or t2==1):
        t1=0
        t2=0
        former.append(y[i])
        latter.append(px11 * x1 ** 2 + px22 * x2 ** 2 + px33 * x3 ** 2 + pt1 * t1 + pt2 * t2 +
            px1 * x1 + px2 * x2 + px3 * x3 + px12 * x1 * x2 + px13 * x1 * x3 + px23 * x2 * x3 +
            px00)


def __sst(y_no_fitting):
    """
    计算SST(total sum of squares) 总平方和
    :param y_no_predicted: List[int] or array[int] 待拟合的y
    :return: 总平方和SST
    """
    y_mean = sum(y_no_fitting) / len(y_no_fitting)
    s_list = [(y - y_mean) ** 2 for y in y_no_fitting]
    sst = sum(s_list)
    return sst
```

```
def __ssr(y_fitting, y_no_fitting):
    """
    计算SSR(regression sum of squares) 回归平方和
    :param y_fitting: List[int] or array[int] 拟合好的y值
    :param y_no_fitting: List[int] or array[int] 待拟合y值
    :return: 回归平方和SSR
    """
    y_mean = sum(y_no_fitting) / len(y_no_fitting)
    s_list = [(y - y_mean) ** 2 for y in y_fitting]
    ssr = sum(s_list)
    return ssr


def __sse(y_fitting, y_no_fitting):
    """
    计算SSE(error sum of squares) 残差平方和
    :param y_fitting: List[int] or array[int] 拟合好的y值
    :param y_no_fitting: List[int] or array[int] 待拟合y值
    :return: 残差平方和SSE
    """
    s_list = [(y_fitting[i] - y_no_fitting[i]) ** 2 for i in range(len(y_fitting))]
    sse = sum(s_list)
    return sse


def goodness_of_fit(y_fitting, y_no_fitting):
    """
    计算拟合优度R^2
    :param y_fitting: List[int] or array[int] 拟合好的y值
    :param y_no_fitting: List[int] or array[int] 待拟合y值
    :return: 拟合优度R^2
    """
    SSR = __ssr(y_fitting, y_no_fitting)
    SST = __sst(y_no_fitting)
    rr = SSR / SST
    return rr


print(goodness_of_fit(y2, y))


print(former)
print(latter)

ans=np.zeros([len(former),2])

ans[:,0]=np.array(former)
ans[:,1]=np.array(latter)
np.savetxt("高钾氧化铜预测.csv",ans,delimiter=',')

print(popt)
```

第二问第一小问逻辑回归

```
# Create SVM classification object
import pandas as pd
from sklearn import metrics
from sklearn.metrics import precision_score, recall_score, f1_score
from sklearn.model_selection import train_test_split
from sklearn import svm, linear_model

x=pd.read_excel("分类表格整理.xlsx")
x=x.values
```

```
y=pd.read_excel("分类表格结果.xlsx")
y=y.values
y=y.flatten()


x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=1,train_size=0.7)


model=linear_model.LogisticRegression(max_iter=1000)
model.fit(x_train,y_train)

y_test_predict=model.predict(x_test)

p_test=precision_score(y_test,y_test_predict)
r_test=recall_score(y_test,y_test_predict)
f1_test=f1_score(y_test,y_test_predict)
print(p_test,r_test,f1_test)

# print(model.coef_)
# print(model.intercept_)

rocy=model.predict_proba(x_test)
# print(rocy)


x3=pd.read_excel("问题3数据.xlsx")
x3=x3.values
pro3pre=model.predict(x3)
print(pro3pre)

print(metrics.log_loss(y_test,y_test_predict))
```

轮廓系数计算

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
from sklearn.cluster import AgglomerativeClustering
from sklearn.preprocessing import MinMaxScaler
from sklearn import metrics


data = pd.read_excel("铅钡筛选后数据.xlsx")

data = data.values

data_array = data[:, 1:5]
model = AgglomerativeClustering(n_clusters=3, linkage='average')
mydata = data_array
y = data[:, 5]
print(model.fit_predict(data_array))

print("轮廓系数铅钡：", metrics.silhouette_score(data_array,model.fit_predict(data_array),
    metric='euclidean'))
```

第二问第三小问敏感性分析

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
from sklearn.cluster import AgglomerativeClustering
from sklearn.preprocessing import MinMaxScaler
```

```
data = pd.read_excel("铅钡筛选后数据.xlsx")

data = data.values

data_array = data[:, 1:5]
model = AgglomerativeClustering(n_clusters=3, linkage='average')
mydata = data_array
y = data[:, 5]
print(model.fit_predict(data_array))
anchor1 = [0, 11, 32]
anchor2 = [36]
anchor3 = [48, 43, 40]

idxlist1 = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24,
    25, 26, 27, 28, 29, 30,
          31, 33, 34]

idxlist2 = [35, 37]

idxlist3 = [38, 39, 41, 42, 44, 45, 46, 47]

# idx=1 #
# idx=2  #提高50% 无变化
# idx=4   #提高50% 无变化
# idx=5   #提高50% 无变化
# idx=21   #提高50% 变化了

# idxlist=np.arange(35)
#
# idxlist=np.delete(idxlist,anchor1)


ylabel = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 2, 2,
        2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]

changesum = 0

# 可调参数
gaibianfudu = 0.9
zhelunzhibiaoxuhao = 0 # 用于选二氧化硅等 0表示二氧化硅 1表示氧化铅 2是五氧化二磷 3是氧化锶

print("种类1")
for idx in idxlist1:
    anchor_of_idx = 0 # 种类
    data = pd.read_excel("铅钡筛选后数据.xlsx")

    data = data.values

    data_array = data[:, 1:5]
    data_array_lat = data_array
    data_array_lat[idx, zhelunzhibiaoxuhao] = data_array_lat[idx, zhelunzhibiaoxuhao] *
        gaibianfudu

    model = AgglomerativeClustering(n_clusters=3, linkage='average')
    latlabel = model.fit_predict(data_array_lat)
    # print(latlabel)

    change = 1

    if (anchor_of_idx == 0):
        flag = 0
        if (latlabel[idx] == latlabel[anchor1[0]]):
```

```
            flag += 1
        if (latlabel[idx] == latlabel[anchor1[1]]):
            flag += 1
        if (latlabel[idx] == latlabel[anchor1[2]]):
            flag += 1
        if (flag >= 2):
            change = 0

    if (anchor_of_idx == 1):
        if (latlabel[idx] == latlabel[anchor2[0]]):
            change = 0

    if (anchor_of_idx == 2):
        flag = 0
        if (latlabel[idx] == latlabel[anchor3[0]]):
            flag += 1
        if (latlabel[idx] == latlabel[anchor3[1]]):
            flag += 1
        if (latlabel[idx] == latlabel[anchor3[2]]):
            flag += 1
        if (flag >= 2):
            change = 0
    changesum += change
    if (change == 1):
        print(idx)

print("种类2")
for idx in idxlist2:
    anchor_of_idx = 1 # 种类
    data = pd.read_excel("铅钡筛选后数据.xlsx")

    data = data.values

    data_array = data[:, 1:5]

    data_array_lat = data_array
    data_array_lat[idx, zhelunzhibiaoxuhao] = data_array_lat[idx, zhelunzhibiaoxuhao] *
        gaibianfudu

    model = AgglomerativeClustering(n_clusters=3, linkage='average')
    latlabel = model.fit_predict(data_array_lat)
    # print(latlabel)

    change = 1

    if (anchor_of_idx == 0):
        flag = 0
        if (latlabel[idx] == latlabel[anchor1[0]]):
            flag += 1
        if (latlabel[idx] == latlabel[anchor1[1]]):
            flag += 1
        if (latlabel[idx] == latlabel[anchor1[2]]):
            flag += 1
        if (flag >= 2):
            change = 0

    if (anchor_of_idx == 1):
        if (latlabel[idx] == latlabel[anchor2[0]]):
            change = 0

    if (anchor_of_idx == 2):
        flag = 0
        if (latlabel[idx] == latlabel[anchor3[0]]):
            flag += 1
```

```
        if (latlabel[idx] == latlabel[anchor3[1]]):
            flag += 1
        if (latlabel[idx] == latlabel[anchor3[2]]):
            flag += 1
        if (flag >= 2):
            change = 0
    changesum += change
    if (change == 1):
        print(idx)

print("种类3")
for idx in idxlist3:
    anchor_of_idx = 2 # 种类
    data = pd.read_excel("铅钡筛选后数据.xlsx")

    data = data.values

    data_array = data[:, 1:5]
    data_array_lat = data_array
    # print(data_array_lat)
    data_array_lat[idx, zhelunzhibiaoxuhao] = data_array_lat[idx, zhelunzhibiaoxuhao] *
        gaibianfudu

    model = AgglomerativeClustering(n_clusters=3, linkage='average')
    latlabel = model.fit_predict(data_array_lat)

    change = 1

    if (anchor_of_idx == 0):
        flag = 0
        if (latlabel[idx] == latlabel[anchor1[0]]):
            flag += 1
        if (latlabel[idx] == latlabel[anchor1[1]]):
            flag += 1
        if (latlabel[idx] == latlabel[anchor1[2]]):
            flag += 1
        if (flag >= 2):
            change = 0

    if (anchor_of_idx == 1):
        if (latlabel[idx] == latlabel[anchor2[0]]):
            change = 0

    if (anchor_of_idx == 2):
        flag = 0
        if (latlabel[idx] == latlabel[anchor3[0]]):
            flag += 1
        if (latlabel[idx] == latlabel[anchor3[1]]):
            flag += 1
        if (latlabel[idx] == latlabel[anchor3[2]]):
            flag += 1
        if (flag >= 2):
            change = 0
    changesum += change
    if (change == 1):
        print(idx)
print("改变量总和", changesum)
```

第三问第一小问玻璃类别分类

```
# Create SVM classification object
import pandas as pd
from sklearn import metrics
from sklearn.metrics import precision_score, recall_score, f1_score
```

```
from sklearn.model_selection import train_test_split
from sklearn import svm, linear_model
import numpy as np

x = pd.read_excel("分类表格整理.xlsx")
x = x.values

y = pd.read_excel("分类表格结果.xlsx")
y = y.values
y = y.flatten()

x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=1, train_size=0.7)

model = linear_model.LogisticRegression(max_iter=1000)
model.fit(x_train, y_train)

y_test_predict = model.predict(x_test)

p_test = precision_score(y_test, y_test_predict)
r_test = recall_score(y_test, y_test_predict)
f1_test = f1_score(y_test, y_test_predict)
print(p_test, r_test, f1_test)

# print(model.coef_)
# print(model.intercept_)

rocy = model.predict_proba(x_test)
# print(rocy)


x3 = pd.read_excel("问题3数据.xlsx")
x3 = x3.values
pro3pre = model.predict(x3)
print(pro3pre)

# print(metrics.log_loss(y_test,y_test_predict))

deltapre_proba = model.predict_proba(x_test)

probabi = np.zeros([21])
for i in range(21):
    probabi[i] = deltapre_proba[i, y_test[i]]

print(metrics.log_loss(y_test, probabi))
```

第三问第二小问模型内部敏感性分析

```
# Create SVM classification object
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn import metrics
from sklearn.metrics import precision_score, recall_score, f1_score
from sklearn.model_selection import train_test_split
from sklearn import svm, linear_model

x = pd.read_excel("分类表格整理.xlsx")
x = x.values

y = pd.read_excel("分类表格结果.xlsx")
y = y.values
y = y.flatten()

x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=1, train_size=0.7)
```

```
cidx = np.linspace(0.000001, 100, 300)
res = []

for t in cidx:
    model = linear_model.LogisticRegression(C=t)
    model.fit(x_train, y_train)
    y_test_predict = model.predict(x_test)
    deltapre_proba=model.predict_proba(x_test)

    probabi=np.zeros([21])
    for i in range(21):
        probabi[i]=deltapre_proba[i,y_test[i]]

    res.append(metrics.log_loss(y_test, probabi))


plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

plt.xlabel('c值序号', fontsize=13)
plt.ylabel('logloss',fontsize=13)
plt.plot(np.arange(300),res)
plt.show()


tidx=np.linspace(0.000001, 100, 300)
rest=[]
for t in tidx:
    model = linear_model.LogisticRegression(tol=t)
    model.fit(x_train, y_train)
    y_test_predict = model.predict(x_test)
    deltapre_proba=model.predict_proba(x_test)

    probabi=np.zeros([21])
    for i in range(21):
        probabi[i]=deltapre_proba[i,y_test[i]]

    rest.append(metrics.log_loss(y_test, probabi))

plt.plot(np.arange(300),rest)
plt.xlabel('tol值序号', fontsize=13)
plt.ylabel('logloss',fontsize=13)
plt.show()


# model=linear_model.LogisticRegression(max_iter=1000)
# model.fit(x_train,y_train)

# y_test_predict = model.predict(x_test)
#
# print(metrics.log_loss(y_test, y_test_predict))

# p_test=precision_score(y_test,y_test_predict)
# r_test=recall_score(y_test,y_test_predict)
# f1_test=f1_score(y_test,y_test_predict)
# print(p_test,r_test,f1_test)

# print(model.coef_)
# print(model.intercept_)

# rocy=model.predict_proba(x_test)
# print(rocy)
```

```
# x3=pd.read_excel("问题3数据.xlsx")
# x3=x3.values
# pro3pre=model.predict(x3)
# print(pro3pre)
```

第三问第二小问氧化铝输入敏感性分析

```
# Create SVM classification object
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
from sklearn import metrics
from sklearn.metrics import precision_score, recall_score, f1_score
from sklearn.model_selection import train_test_split
from sklearn import svm, linear_model

x = pd.read_excel("分类表格整理.xlsx")
x = x.values

y = pd.read_excel("分类表格结果.xlsx")
y = y.values
y = y.flatten()

x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=1, train_size=0.7)

model = linear_model.LogisticRegression(max_iter=1000)
model.fit(x_train, y_train)

# y_test_predict=model.predict(x_test)

# p_test=precision_score(y_test,y_test_predict)
# r_test=recall_score(y_test,y_test_predict)
# f1_test=f1_score(y_test,y_test_predict)
# print(p_test,r_test,f1_test)

# print(model.coef_)
# print(model.intercept_)

# rocy=model.predict_proba(x_test)
# print(rocy)


x3 = pd.read_excel("问题3数据.xlsx")
x3 = x3.values
# pro3pre = model.predict(x3)
# print(pro3pre)
#
# print(metrics.log_loss(y_test,y_test_predict))


truelabel = [0, 1, 1, 1, 1, 0, 0, 1]

"""
二氧化硅 12
氧化钠 0.24
氧化钾 0.168
氧化钙 0.702
氧化镁
氧化铝(Al2O3)
氧化铁(Fe2O3)
氧化铜(CuO)
氧化铅(PbO)
氧化钡(BaO)
```

```
五氧化二磷(P2O5)
氧化锶(SrO)
氧化锡(SnO2)
二氧化硫(SO2)

"""

idx = 6
zengliang = 12

bound=100

deltalist = np.linspace(0, bound, bound)

losslist = []

for delta in deltalist:
    x3 = pd.read_excel("问题3数据.xlsx")
    x3 = x3.values
    myx = x3
    myx[:, idx] = myx[:, idx] + delta
    deltapre = model.predict(myx)
    deltapre_proba = model.predict_proba(myx)

    probabi = np.zeros([8])
    for i in range(8):
        probabi[i] = deltapre_proba[i, truelabel[i]]

    losslist.append(metrics.log_loss(truelabel, probabi))

plt.plot(deltalist,losslist)

plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

plt.xlabel('氧化铝增量值', fontsize=13)
plt.ylabel('logloss',fontsize=13)
plt.show()
```

第四问第一小问皮尔逊相关系数检验

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import spearmanr, pearsonr

plt.rcParams['font.sans-serif'] = ['FangSong']
plt.rcParams['axes.unicode_minus'] = False
z = pd.read_excel("铅钡总表.xlsx")
z = z.values

data = z[:, :]

res = pd.DataFrame(data, columns=[
    '二氧化硅', '氧化钠', '氧化钾', '氧化钙', '氧化镁', '氧化铝', '氧化铁', '氧化铜', '氧化铅',
    '氧化钡', '五氧化二磷', '氧化锶', '氧化锡', '二氧化硫'], dtype=np.float)

corr = res.corr(method='pearson')

sns.heatmap(corr, annot=True, vmax=1, vmin=-1, xticklabels=True, yticklabels=True,
            square=True,
                cmap="RdYlGn",annot_kws={"fontsize":7}).get_figure().savefig("temp.png",dpi=500,bbox_inches
                = 'tight')
```

```
# plt.show()


name = ['二氧化硅', '氧化钠', '氧化钾', '氧化钙', '氧化镁', '氧化铝', '氧化铁', '氧化铜',
    '氧化铅',
      '氧化钡', '五氧化二磷', '氧化锶', '氧化锡', '二氧化硫']

# print(pearsonr(res['过滤阻力'], res['透气性']))

matriccor=np.zeros([14,14])
for n in range(14):
    for m in range(n + 1, 14):
        # print(name[n],name[m])
        # print(pearsonr(res[name[n]], res[name[m]]))
        _,pvalue=pearsonr(res[name[n]], res[name[m]])
        matriccor[n,m]=pvalue

np.savetxt("铅钡热图p值.csv",matriccor,delimiter=',')
```

第四问第二小问 z 检验

```
import math

import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import spearmanr, pearsonr
from scipy.stats import norm

plt.rcParams['font.sans-serif'] = ['FangSong']
plt.rcParams['axes.unicode_minus'] = False
z = pd.read_excel("高钾总表.xlsx")
z = z.values

data = z[:, :]

res = pd.DataFrame(data, columns=[
    '二氧化硅', '氧化钠', '氧化钾', '氧化钙', '氧化镁', '氧化铝', '氧化铁', '氧化铜', '氧化铅',
    '氧化钡', '五氧化二磷', '氧化锶', '氧化锡', '二氧化硫'], dtype=np.float)

corrgaojia = res.corr(method='pearson')
corrgaojia = corrgaojia.values
z = pd.read_excel("铅钡总表.xlsx")
z = z.values

data = z[:, :]

res = pd.DataFrame(data, columns=[
    '二氧化硅', '氧化钠', '氧化钾', '氧化钙', '氧化镁', '氧化铝', '氧化铁', '氧化铜', '氧化铅',
    '氧化钡', '五氧化二磷', '氧化锶', '氧化锡', '二氧化硫'], dtype=np.float)

corrqianbei = res.corr(method='pearson')

corrqianbei = corrqianbei.values
zpvalue = np.zeros([14, 14])

for i in range(14):
    for j in range(14):
        if (i != j):
            gaojia = corrgaojia[i, j]
            qianbei = corrqianbei[i, j]
```

```
            r1 = gaojia
            n1 = 18
            r2 = qianbei
            n2 = 49

            z1 = 0.5 * math.log((1 + r1) / (1 - r1))
            z2 = 0.5 * math.log((1 + r2) / (1 - r2))

            ddiff = z1 - z2
            SEddiff = math.sqrt(1 / (n1 - 3) + 1 / (n2 - 3))

            zvalue = ddiff / SEddiff
            pvalue = 2 * (1 - norm.cdf(abs(zvalue)))
            zpvalue[i, j] = pvalue
np.savetxt("z检验p值结果.csv",zpvalue,delimiter=',')
```